

Rapid modelling of the target tracking loop of a passive IR guided SAM, using artificial neural systems



by Thomas Jones

Thesis presented in partial fulfilment of the requirements for the degree
of Master of Science in Engineering at the University of Stellenbosch.

December 1999

Supervisor: Prof. J.J. du Plessis

DECLARATION

I, the undersigned, hereby declare that the work in this thesis is my own original work and has not previously in its entirety or in part been submitted at any university for a degree.

T. Jones

ABSTRACT

This thesis deals with the modelling of the target tracking system of a specific passive infrared guided surface to air missile (SAM). Modelling is accomplished by making use of rapid modelling principles and procedures in order to establish whether an accurate target tracking model may be constructed whilst at the same time keeping construction time to an absolute minimum. The specific rapid modelling techniques used during the course of the study centre around the application of hybrid artificial neural structures in an attempt to adequately represent the missile's target tracking control system.

Various criteria are developed, whereby the target tracking control system features and structures of this type of missile may be extracted through minimal analysis and disassembly of the missile. Time requirements are thereby minimised. The tracking loop is characterised by making use of signal injection into the missile electronics. The eventual construction of a model is discussed as a progression, beginning with the development of a fully artificial neural structure, and developing towards a fully optimised hybrid neural model.

The final model's eventual output results are analysed according to error energy, statistical principles and frequency content. Finally, a comprehensive hardware in loop (HIL) simulation is completed, whereby the model replaces the missile's target tracking electronics under real-time physical test conditions, while tracking a true infrared target. The model's response compares favourably to the true missile's behavior during HIL proceedings. Together with the favourable theoretical analysis of the model, this illustrates that rapid modelling of the tracking system may be successfully completed when applying the techniques developed in this thesis.

OPSOMMING

Hierdie tesis handel oor die modellering van die teiken-volglus van 'n passiewe infrarooi-geleide grond-tot-lug missiel. Modellering word afgehandel deur gebruik te maak van sogenaamde “vinnige modellerings tegnieke”, ten einde te bepaal of 'n akkurate model van so 'n volgstelsel bewerkstellig kan word terwyl die modellerings prosedure besonder spoedig afgehandel word. Die spesifieke modellerings prosedure waarvan gebruik gemaak word tydens hierdie studie, fokus op die gebruik van hibriede kunsmatige neurale strukture.

Verskeie kriteria word ontwikkel waarmee die kenmerke en strukturele samestelling van verwante missiele se beheerstelsels bepaal kan word sonder om veel detail analise of demontering van die missiel te vereis, ten einde die modelleringstyd te minimeer. Die volglus word gekarakteriseer deur gebruik te maak van eksterne sein aanleg. Die konstruksie van die model word bespreek as 'n progressie, beginnende by die ontwikkeling van 'n ten volle neurale model totdat 'n ten volle geoptimeerde hibriede neurale struktuur uiteindelik ontwikkel word.

Die uiteindelijke model se uittree-resultate word geanaliseer volgens foutenergie, statistiese metodes en frekwensie inhoud. Laastens word 'n volledige hardware in die lus (HIL) simulاسie verrig waartydens die missiel se volg-elektronika vervang word deur die hibriede model, onder intydse fisiese toestande, terwyl 'n ware infrarooi teiken gevolg word. Die model se respons vergelyk goed met die ware missiel se gedrag tydens HIL simulاسie. Tesame met die voorgaande akkurate teoretiese resultate, bewys dit dat vinnige modellering van die volglus wel bewerkstellig kan word deur gebruik te maak van die tegnieke wat in hierdie tesis ontwikkel en toegepas word.

ACKNOWLEDGEMENTS

At the University of Stellenbosch:

Prof. J.J. du Plessis, whose leadership, guidance, knowledge and experience was vital during the course of this project.

Dr. D.M. Weber, for his assistance regarding neural network structures and optimised neural computing.

Mr. N. Goosen, for his clear thoughts on the subjects of rule extraction and neural network training.

Mr. W Croukamp, for his technical expertise and the development of a rate table fitting for the housing of the missile's gyroscope.

Mr. H. Grobler, for his assistance regarding software issues and software support.



At Aerotek, CSIR:

Aerotek, CSIR, for financial and logistical backing of this project.

Dr. J.H.S. Roodt, for providing that proverbial shove back to reality and practicality, especially with regard to neural networks, and for his efforts regarding the financial and logistical backing of this project.

Mr. H.J. Theron, for sharing his intimate knowledge of missile operation with me, for his assistance during the HIL tests and for keeping calm while I made the mistakes.

Mr. M.J.U. du Plooy, for his knowledge regarding missile simulation hardware and practical missile interfacing issues, and for only chasing me from his office once.

TABLE OF CONTENTS

1. Introduction	11
1.1 Background	11
1.2 Project Goals (Problem Statement)	12
1.3 Literature study	13
1.4 Thesis Structure	14
2. SAM specifications	15
2.1 Seeker.....	15
2.1.1 Physical Characteristics	15
2.1.2 Optics	15
2.1.3 Seeker Electronics	16
3. Capturing the essence of a SAM	20
3.1 Locations for injection/recording	20
3.2 Creation of suitable injection material	21
3.2.1 Criteria for signal injection	21
3.2.2 Signal generation/Model of optics	22
4. Signal injection and recording	27
4.1 Hardware configuration	27
4.2 Software configuration	29
4.3 Injection series I/O grouping	30
5. Relevant modelling techniques	35
5.1 Gain vs. Compromise.....	35
5.2 Rapid modelling approaches attempted	36
5.2.1 Solo Network	36
5.2.2 Generic pre-processing with sub-sampling to reduced network.....	46
5.2.3 Sub-sampled, optimised structural pre-processing to a reduced network....	58
6. MODEL Implementation and HIL simulation.....	81
6.1 Introduction.....	81
6.2 Model Implementation Hardware	82
6.2.1 Missile front end	82
6.2.2 Missile electronics.....	83
6.2.3 A/D and D/A hardware	84
6.2.4 Target simulator set-up	85
6.3 Software	86
6.4 Target tracking results.....	89
6.4.1 Stationary target	89
6.4.2 Step response.....	92
6.4.3 Rate tracking response	94
6.4.4 In search of trouble	96

6.4.5	HIL validation of the re-optimised solution	102
7.	Project evaluation and conclusions.....	105
7.1	Accomplishments.....	105
7.1.1	Signal injection	105
7.1.2	Modelling	105
7.1.3	Model implementation and HIL.....	106
7.1.4	Gyroscopic modelling	106
7.2	Limitations	106
7.3	Possible improvements	107
7.4	Final remarks	108
8.	References	109
9.	Appendix A: Gyroscopic modelling	111
A.1	Coil output vs. Look angle calibration	111
A.2	Gyro signal injection response measurements.....	112
A.3	Model optimisation.....	114
10.	APPENDIX B: Software catalogue	116
B.1	Optics/Reticle emulation software (<i>Matlab</i>).....	116
B.2	Injection/Recording software (<i>Pascal</i>)	118
B.3	HIL Simulation software (GCC).....	122
	Main simulation file: SNNS20d.c.....	122
	Protected mode timer interrupt header: Timer.h.....	125
	Protected mode timer interrupt: Timer.s	126
	Neural network function header: SNNS20d.h (GEN. BY SNNS2C).....	128
	Neural network function: SNNS20d.c (GEN. BY SNNS2C)	129
B.4	Gyro characterisation: Gyro2.PAS (<i>Pascal</i>).....	137
11.	APPENDIX C: A/D and D/A converter modelling	142
C.1	A/D converter response measurement.....	142
C.2	A/D converter modelling	143

LIST OF FIGURES

Figure 1	SAM FM Reticle	16
Figure 2	Navigational Electronics	17
Figure 3	Missile head coil assembly	19
Figure 4	C130 IR image with vertical clutter	23
Figure 5	Typical IR irradiance envelope at detector during 10s of flight	26
Figure 6	Hardware signal injection/recording set-up	27
Figure 7	Typical RMS error progression during training using four different algorithms	40
Figure 8	The sigmoid training function	42
Figure 9	A 300x15x1 Multi-layer Perceptron Network	44
Figure 10	Network output response to training data	45
Figure 11	Typical precession signal frequency content	48
Figure 12	Generic preamplifier frequency response	50
Figure 13	Harmonic distribution because of filtering and saturation	52
Figure 14	Generic precession amplifier frequency response	53
Figure 15	Typical pre-processing output vs. recorded missile output	54
Figure 16	MSE training progress of neural network with generic pre-processing	56
Figure 17	Response of network with generic pre-processing to training data windows	57
Figure 18	Pre-processing model optimisation structure	60
Figure 19	Pre-processing demodulator structure	62
Figure 20	Pre-processing precession amplifier structure	63
Figure 21	Typical pre-processing training data output results	64
Figure 22	Typical pre-processing test data results	65
Figure 23	Missile response vs. pre-processing response for Oryx rear aspect view	66
Figure 24	Z-transformed pre-processing structure	68
Figure 25	Final pre-processing structure prior to neural network training	68
Figure 26	Typical final Z-plane pre-processing results	70
Figure 27	Respective neural network training progress	71
Figure 28	Optimised pre-processing & neural network combination training outputs	75
Figure 29	Comparative training data frequency response of model and missile	76
Figure 30	Typical optimised pre-processing and neural network combination test outputs	78
Figure 31	Series s2 and s4 relative responses after gyro and precession LPF action	80
Figure 32	Basic precession drive schematic	84
Figure 33	HIL Simulation set-up block diagram	86
Figure 34	Stationary target HIL simulation vs. Missile relative LOS response	91
Figure 35	HIL simulation vs. missile, stationary target tracking pattern	91
Figure 36	HIL simulation vs. Missile relative LOS target step response	93
Figure 37	HIL simulation vs. Missile relative LOS target step response transient ZOOM	93
Figure 38	HIL simulation vs. Missile relative LOS target rate tracking response	95
Figure 39	Open loop pre-processing model's responses sinusoid frequency inputs	98
Figure 40	RE-OPTIMISED open loop pre-processing model's sinusoid input response	101
Figure 41	Typical re-optimised final Z-plane pre-processing results, ref. Figure 26	102
Figure 42	Re-optimised model's HIL rate tracking response	103
Figure 43	Gyro Look Angle vs. Cage coil voltage calibration	112
Figure 44	Typical precession signal and corresponding cage coil response	113
Figure 45	Gyro model optimisation structure	114
Figure 46	A/D converter response measurement	142
Figure 47	A/D converter model optimisation	143

LIST OF TABLES

Table 1 Missile IR detector injection series 31

Table 2 Injection series distribution amongst training and test sets 33

Table 3 Deterministic modelling vs. Rapid modelling 35

LIST OF ABBREVIATIONS AND ACRONYMS

AA	Air to Air / Anti-aliasing (see context)
AAF	Anti-aliasing Filter
AC	Alternating Current
A/D	Analogue to Digital
Aerotek	Division of Manufacturing and Aeronautical Systems Technology
AM	Amplitude Modulation
BP	Bandpass
BPF	Bandpass Filter
BW	Bandwidth
CSFM	Conical Scan Frequency Modulation
CSIR	Council for Scientific and Industrial Research
D/A	Digital to Analogue
DC	Direct Current
DPMI	Dynamic Protected Mode Interface
FIR	Finite Impulse Response
FM	Frequency Modulation
FOV	Field of View
gyro	gyroscope
HIL	Hardware In Loop
IIR	Infinite Impulse Response
InSb	Indium-Antimonide
I/O	Input/Output
IR	Infrared
LCR	Inductance-Capacitance-Resistance
LOS	Line of Sight
LPF	Lowpass Filter
MPa	Mega-Pascal
MSE	Mean Square Error
NC	Navigational Constant
NEFD	Noise Equivalent Flux Density

Op-Amp	Operational Amplifier
PC	Personal Computer
Preamp	Preamplifier
PropNav	Proportional Navigation
PWM	Pulse Width Modulation
RAM	Random Access Memory
REP	Relative Error Percentage
RMS	Root Mean Square
SAM	Surface to Air Missile
SCG	Scaled Conjugate Gradient
SISO	Single Input Single Output
SID	System Identification
SM	System Modelling
SNNS	Stuttgart Neural Network Simulator
SSE	Sum Square Error
TAG	Target Adaptive Guidance
ZOH	Zero-Order Hold

CHAPTER 1

INTRODUCTION

1.1 Background

The author became involved with Aerotek (CSIR) in December 1996, employed as part of a program to develop countermeasures against known types of Surface to Air Missiles (SAMs) available on the African continent. The focus of this thesis is the ascertaining of the safety of all military and commercial aircraft during flights over Africa.

A substantial part of countermeasure development is comprised of missile modelling, since these models form part of the creation and validation platform of countermeasure systems. This thesis covers the development of the target tracking loop model for one such passive IR guided missile. The target tracking loop is the main control/sensing loop influenced by countermeasures (CM). Most countermeasure tests involve only validation that the tracking loop has lost track of the target position, since a target miss-hit is virtually assured once a missile loses its target tracking capability¹.

The author has previously developed full control, tracking and flight models for two different types of missiles, by disassembling and modelling various parts of these missiles. Missile modelling requirements have started to evolve in a different direction recently however. This is known as “Rapid Modelling,” where time, manpower and technological constraints necessitate the development of accurate missile models in the shortest possible time. The time frame for the development of a 1970’s type IR guided SAM model (such as for the well-known Russian series) would, for instance, need to be limited to three or four months. Such models took years to develop previously, as full disassembly and analysis of a missile proves a laborious task.

¹ Providing that the tracking loss occurs early enough and that the target remains in non-rectilinear or non-stationary flight.

Recent developments in the field of artificial neural networks and computing technology, allow exploration of a different means of modelling, making use of neural based structures in order to represent complex data transfer patterns. A missile tracking loop can be modelled as such a complex data transfer pattern, where a target scenario image may for instance be converted to a gyroscopic precession signal. The problem cannot however be tackled by simply “throwing” a neural network at the problem and expecting a solution. Serious constraints regarding neural network optimisation/training abilities and modelling accuracy exist which require prior resolution.

This thesis analyses different means of neural-based modelling, and suggests various criteria for the development of such systems, both for this specific missile, and for generic missiles falling under the banner of passive IR guidance. A neural based model is developed for the given missile and is verified under HIL conditions; where the missile’s tracking gyro, optics, IR detector and true IR sources are included.

While most work of this type currently under way at Aerotek focuses on the development of HIL type systems for countermeasure tests, this thesis provides a bridge between the full software approach and HIL countermeasure validation. Removing the electronics of the missile gyroscope and replacing them with the neural-based model is an instructive experiment that paves the way for full HIL simulation of complex target scenes. The model developed in this thesis, however, is to be used mostly for full software modelling, where even the gyro and the missile optics have been turned into a discrete model. Appendix A discusses the construction of such a gyro model while Chapter 3 is concerned with the modelling of the missile optics. Eventually, this thesis will be developed into a stand-alone project, where a client may require the development of a full software model of a missile, without that client having the need for any equipment other than a personal computer. They would typically provide two specimens of the missile in question and receive the required model in the shortest possible time.

1.2 Project Goals (Problem Statement)

- Various artificial neural system-based structures will be evaluated in terms of their suitability to model the control electronics of a specific analogue type passive IR guided missile tracking loop.

- A simple missile signal injection, recording and characterisation system will be developed, making use of as little prior knowledge of the missile as possible. This forms part of the “Rapid Modelling” philosophy.
- Criteria will be developed regarding the choice of missile signal injection material for typical passive IR guided seekers. The aim is to use the missile’s response to such inputs for neural network training purposes. Such injection material will then be developed and injected into the missile tracking system, while recording the missile’s response.
- One neural-based model will be constructed and its accuracy will be verified using additional non-training test data.
- A HIL test will be conducted to validate the model’s behaviour when physically used to replace the target tracking control electronics within the missile.
- The missile gyroscope and reticle-optics may be modelled as an additional step towards full software implementation of the tracking loop.

1.3 Literature study

The material handled within this thesis covers a wide range of topics related to control systems, signal processing and IR target tracking. An indication of some of the more important sources follows:

- The literature sources listed after the conclusion of the thesis include papers on reticle development and analysis that are essential to the understanding of the purpose of various reticle shapes and operational designs, including stationary and spinning FM reticles. Driggers et al (1991) and Chao et al (1988a) (1988b), provide further information regarding this.
- Some references are made to the B.Eng final year thesis of the author. (Jones, 1997) A copy of this report is available from the Department of Electronic Engineering of the University of Stellenbosch. The vast differences in the modelling approaches can be seen. When comparing the two documents.
- Various neural network texts essential to this project are listed. These are a selection of the actual texts used, but are the most instrumental in approaching the modelling problem. Lin et al (1996), Zurada (1992) and Lippmann (1987) prove informative sources regarding this material.

- Various texts are included that focus on aeronautical (esp. missile) control systems. Understanding the principles of proportional navigation and missile guidance explained in Zarchan (1990) is essential to the comprehension of this thesis. Jones (1997) and Bryson (1994) also aid in this.

1.4 Thesis Structure

The thesis structure tracks the development of the project in a chronological fashion. The missile specifications provide a starting point, working through the reticle modelling, signal injection systems and neural network structures, and completing the project with the HIL simulation.

The aim is to represent the thesis as a chain of events which eventually lead to the results and conclusions laid out in Chapter 7. This style of writing will highlight the motivations for the decisions made and the criteria developed. Chapters may be read as separate entities, but this is not encouraged.

Following the reference of sources the appendices are included as additional chapters. Appendix A provides a summary of the development of a model of the missile's gyroscope. The experiment is not intended to be repeatable from this discussion, and is only listed for the sake of the results obtained. Much of the set-up is considered to be of a sensitive or classified nature, as this was the first accurate attempt at modelling the gyroscope. For security reasons the exact name of the missile and sensitive missile specifications and information are omitted. It is unfortunate that some information has to be presented with less clarity than would have been preferred, but this is an understandable precaution.

Appendix B contains a collection of the important software written during the course of the project.

Appendix C contains a summary of the characterisation and modelling of the A/D converter used throughout the project.

CHAPTER 2

SAM SPECIFICATIONS

This chapter considers the technical specifications of the missile in question. It should be kept in mind that much of the data in this chapter is general to various different types of SAMs. Only some of the data supplied serves as a requirement for rapid modelling. As this is an initial attempt at rapid modelling, it could be useful to compare the amount of known data with the amount of necessary data to construct the model.

It is important, however, not to make use of a priori information when deciding on modelling issues, unless clear statement is made that such information should be determined before rapid modelling is started. The aim here is to find a compromise between necessary analysis of the missile, which takes valuable time, and construction of a sufficiently accurate model. The following information also serves to illustrate as far as possible the environment in which the modelling is done, since some of the techniques and compromises that are to be made may not necessarily hold for different types of SAMs.

2.1 Seeker

2.1.1 Physical Characteristics

The missile is approximately 1.5m in length and 70 mm in diameter. It has a mass of 10 kg before launch, with initial centre of gravity located approximately 70cm from the nose. A tail fin/canard aerodynamic configuration is used, whereby directional control is provided by a pair of canards mounted on the seeker. These are pulse width modulated (PWM) as the missile rolls about its centre axis. Stabilisation and roll are provided by four canted fins located at the rear extremity of the missile. All control surfaces fold in so that the missile fits into its launch tube. Propulsion is provided by a two stage solid propellant motor, which accelerates the missile to an approximate sustained velocity of Mach 1.8.

2.1.2 Optics

The seeker is a passive infrared tracker. Cassegrain optics collect and direct infrared energy onto a cooled indium antimonide detector, which operates within the 3.5 to 5.0 μm IR band. Operation in this region of the infrared spectrum provides the SAM with good low-altitude and

head-on capabilities. A conical scan frequency modulation (CSFM) technique is employed for target discrimination. The FM encoding is done by the reticle illustrated in Figure 1. The overall design is effective against cluttered backgrounds (such as clouds or terrain), and is inherently more difficult to defeat with active infrared jammers than previous SAMs of the same configuration.

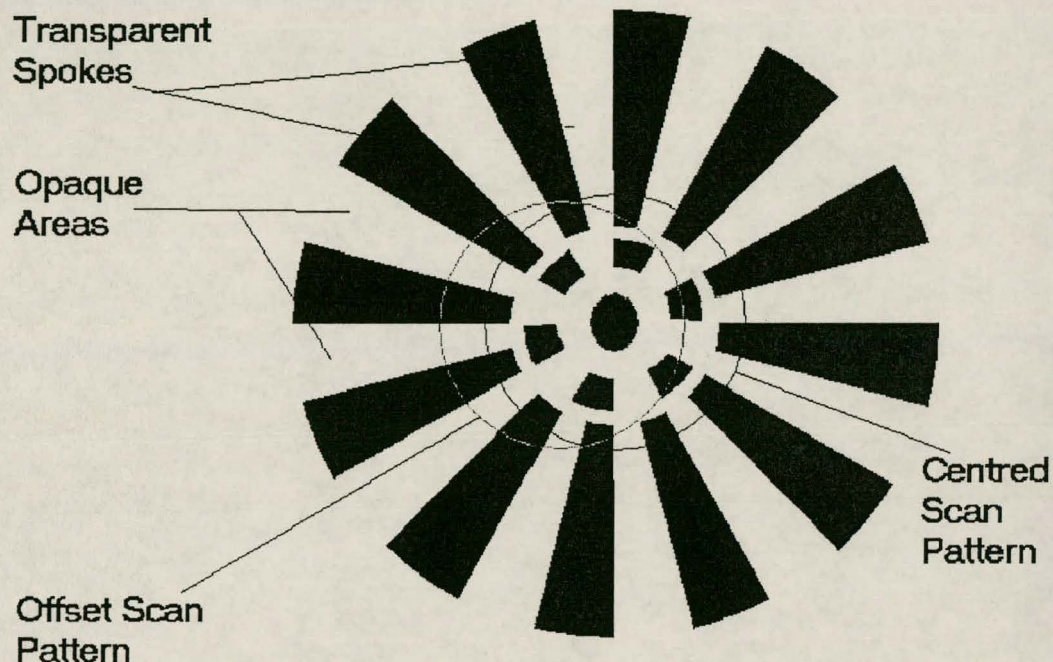


Figure 1 SAM FM Reticle

Figure 1 illustrates twelve transparent spokes across which the conical scan is completed. The six central spokes assist with target focal point resolution when the scan crosses close to the centre of the stationary reticle. None of the parameters of this figure are precisely scaled, and the two indicated target scan patterns are only an illustration of the scan pattern that would result from a point source target in one gyro revolution.

2.1.3 Seeker Electronics

2.1.3.1 Tracking system

Target tracking is accomplished by decoding the CSFM information provided electronically by the IR detector. A bandpass preamplifier with centre frequency at the CSFM carrier frequency removes unwanted information from the detector signal and amplifies it to a partly saturated 20Vpp FM signal. (Saturation occurs around the preamplifier centre frequency only.) The FM signal is then demodulated by the so-called “position amplifier”. The position amplifier

consists of a bandstop filter with a decidedly linear differential flank at the FM carrier frequency, acting as differentiator. This is followed by a rectifier circuit and a first order lowpass filter which removes some of the unwanted high frequency mirror resulting from demodulation. The demodulated signal is then filtered once more by a higher order bandpass filter known as the precession amplifier (and driver), which removes any remaining high and low frequency components above and below the gyro spin frequency. The resulting “precession signal”, is then amplified by a precession amplifier which feeds the signal directly to coils surrounding the gyroscope. In this way, the north/south magnetised gyroscope is steered to point directly towards the target. The upper half of Figure 2 gives a graphic illustration of these circuits.

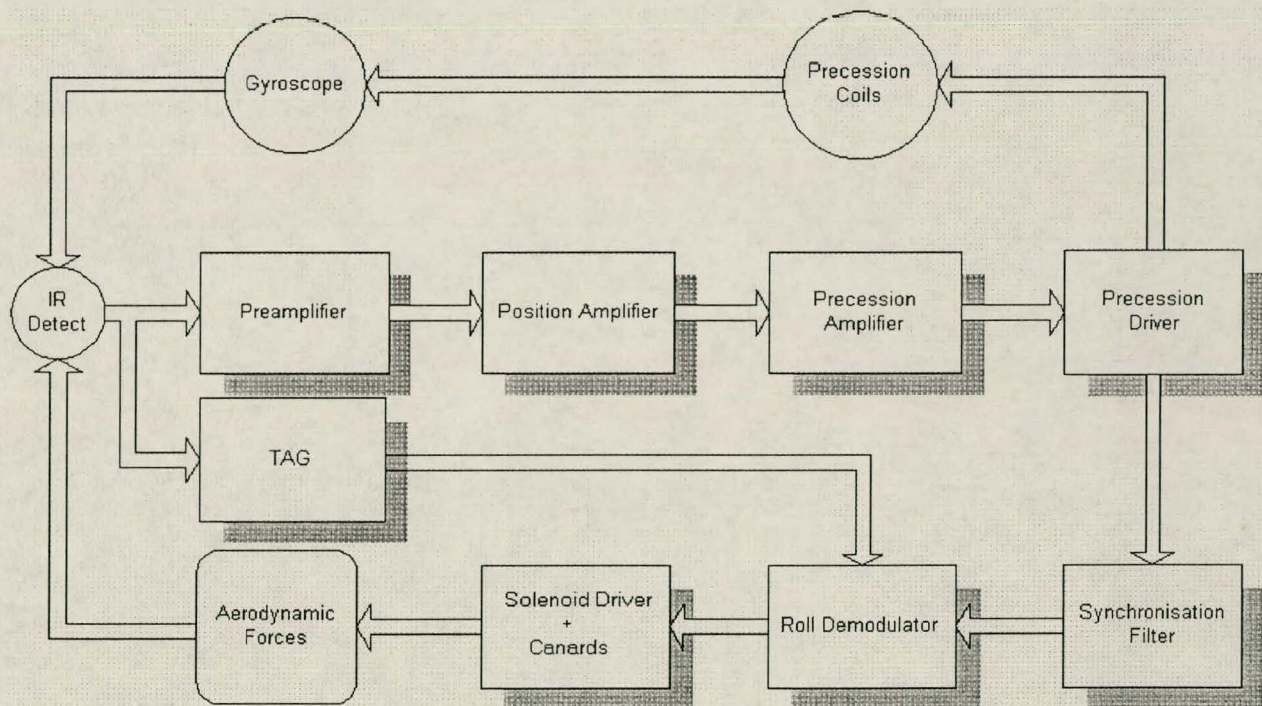


Figure 2 Navigational Electronics

2.1.3.2 Steering system

In order to steer the missile, the synchronised precession signal is mixed with the wing demodulation reference coil's output. After it has been LPF'ed the resulting signal has a frequency equal to the body roll rate, and an amplitude scaling relative to the precession signal's amplitude. Phase information (indicating target direction) is also obtained from the precession signal. This signal is then fed to a squaring amplifier and used to switch the gas solenoid valves that toggle the canard deflection angle. As the canards have no intermediate

deflection angles other than being fully “up” or “down,” a relatively high frequency and low amplitude dither is added to the roll demodulator output before squaring. This applies higher frequency PWM control to the canards instead of the usual “bang-bang” control when the precession signal has a small amplitude. (This indicates that the missile is flying close to its required course and PWM canard control completes the final navigational fine tuning.)

Figure 2 illustrates the steering system described above, together with the tracking system discussed in 2.1.3.1. The resulting type of navigation is known as Proportional Navigation (PropNav), where the rate of change of the missile heading (via the canard deflection) is directly proportional to the rate of change of the line of sight from the missile to the target (represented by the precession signal).

Additional information:

The synchronisation amplifier/filter checks that the precession signal frequency is equal to the gyro spin rate and attenuates it if it is not. By amplifying the precession signal before it is fed to the roll demodulator it also serves as a loop gain switch which increases the so-called Navigation Constant (NC) when the gyro look angle with respect to the body exceeds 10 degrees. A Target Adaptive Guidance (TAG) system comes into operation during the very last stages of flight when the IR detector saturates and the target focal point overlaps with more than one spoke of the reticle in Figure 1. It applies final moment flight correction when the detector signal is outside the preamplifier frequency range (because of focal point/spoke overlaps) and applies exhaust plume correction in order to ensure fuselage impact with the target. (If this is not done, then the missile will fly through the target’s exhaust plume, which is the target of its initial tracking.) Some of the mechanisms of the TAG circuit are explained in Jones (1997), but further analysis of its exact operation is required.

2.1.3.3 Gyroscopic spin regulation

Gyro spin regulation is accomplished by three circuits: The spin regulator, spin amplifier and spin driver. The regulator makes use of signals resulting from the so-called lambda (cage) and reference coils. The reference coils are twisted at right angles to the caged gyro spin plane, thereby collecting spin rate information relative to missile body rotation. The lambda coils are aligned with the caged gyro rotation plane to collect gyro look angle information relative to the missile heading and gyro spin rate information relative to the missile’s inertial frame of reference (or the horizon). The gyro rotation rate is then regulated to 100 Hz with respect to

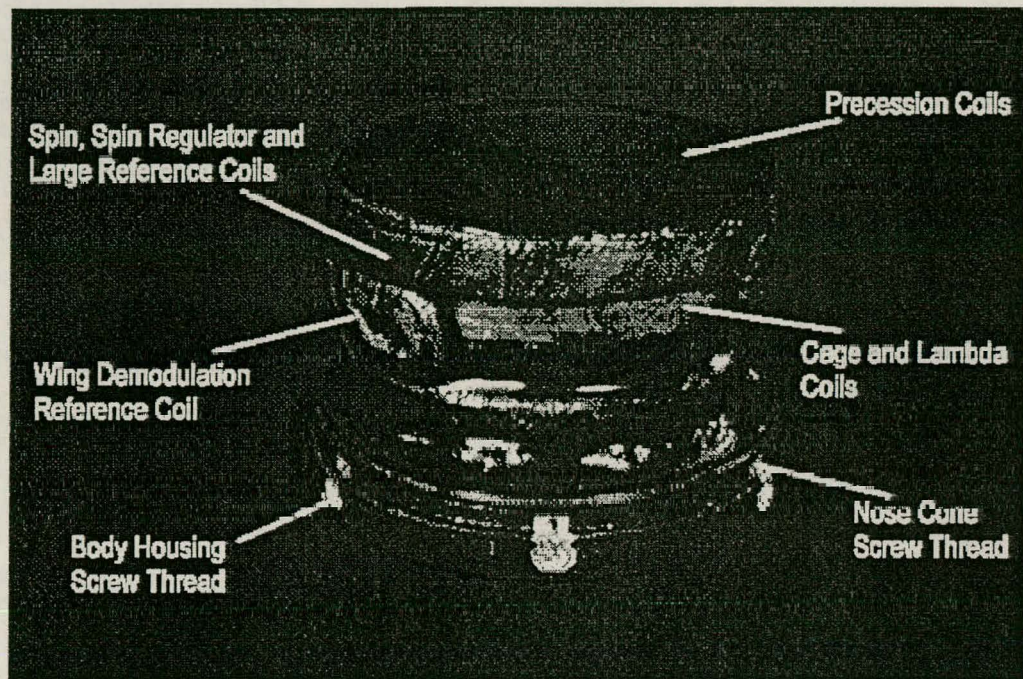


Figure 3 Missile head coil assembly

its inertial frame of reference via the spin coils of the head coil assembly. Simple tests indicate that when a step response is required, the spin control mechanism has a control time constant of 10 to 15 times faster than the average gyro time constant. This fact, together with the robustness of the gyro and the minimal gimbal friction that is exerted on a new gyro, allows the gyro rotation rate to be modelled as a constant throughout this thesis. Figure 3 provides a clear illustration of the missile head coil assembly which surrounds the gyroscope. (In Figure 3 the gyro/radiometer would face upwards, fitting within the blackened cavity.)

CHAPTER 3

CAPTURING THE ESSENCE OF A SAM

Every modelling approach in this study has one aspect in common: their aim is to capture the characteristics of the specific SAM being modelled. In order to be able to fit a model to these characteristics however, it is necessary to obtain these characteristics from the SAM itself. This is done by capturing the response of the missile tracking loop to certain very carefully chosen stimuli. Injecting signals into the missile, and recording its response, forms the basis of data on which the models are to be fitted. Much care has been taken with the choice of recording and injection points, as with the manner in which the injection is accomplished.

3.1 Locations for injection/recording

Several criteria need to be taken into account before the locations for injection/recording can be chosen:

- Other than its natural response to a relevant input signal all coupling to the missile must have as **little influence on its operation** as possible.
- Before being able to commence with signal injection, the coupling should be to parts of the missile that are **easily recognisable** and **easily reached**, to ensure as little analysis of the electronics as possible.
- Injection should take place at a point where very few of the missile characteristics have already had an influence (i.e. as close to the missile's own point of tracking information retrieval as possible).
- Recording should take place as close to the true output of the system as possible (i.e. after most or all of the missile subsystems have played their part).

After these points have been considered the injection point is chosen as the IR detector signal directly before the preamplifier in Figure 2. This is the first known electrical input to the missile and therefore the most basic, unadulterated input, only the optics and detector have played any influential part. Before this point. Unfortunately, being only a few nano-Watts, the input signal is very small under normal operating conditions.²

² This problem is overcome in section 4.1.

The recording access point is chosen to be the so-called “ $\frac{4}{7}\sigma$ ” output of the missile. This point is easily accessible on the outside umbilical plug of the missile, as it is fed to the missile launcher tube for caging purposes before launch takes place. The available signal represents a linearly scaled version of the precession driver’s output. Such a signal can unfortunately only be found after careful analysis of the missile. It serves only as a known acceptable substitute for the precession coil output itself. This output can be found through location of the precession coils in the head coil assembly, and the surveillance of the electronics’ output to these coils. The methods of injection and recording are covered in Chapter 4.

3.2 Creation of suitable injection material

3.2.1 Criteria for signal injection

The criteria requiring consideration for signal injection are as follows:

- The missile typically receives CSFM inputs, resulting from the interaction between the optics, the reticle and the IR detector. Typical characteristics within the missile’s operational parameters are most frequently revealed when typical CSFM signals are injected.
- The entire frequency range that might be received by the missile under normal operational conditions should be covered by the input signals.
- Certain boundary condition inputs help to define the response of the missile to basic signals (such as DC levels, fixed frequency carriers with no FM and carriers with signals of fixed frequency and amplitude FM’ed onto them).
- Most model optimisation algorithms employ error energy minimisation techniques. This implies that the number of input signals within a certain range determine how well the model is fit to that range (e.g. a model fit to the response to 10 seconds of FM at a 1.2 kHz carrier and 2 seconds of FM at 1.4kHz would be more likely to end up being a more accurate model for FM surrounding 1.2kHz than 1.4kHz). It is therefore important to include injection signals that fall only within very specific ranges of operation in order to add these responses to the model optimisation. This is required if it is found that initial modelling does not fit the ranges with sufficient precision.

- In keeping with the previous criterion, there must be a limit to the number of criteria within certain boundary ranges. If there is not, the optimisation routines may force the model to fit only within these frequently activated ranges.
- Any known non-linearities or multiple mode operations, such as gain scheduling, should be excited if at all possible.
- Amplitude ranges of inputs should be representative of typical inputs to the missile.
- Any possible non-linearities within typical inputs (such as saturation) should feature in injection signals.

3.2.2 Signal generation/Model of optics

In order to continue it is necessary to generate injection signals in keeping with 3.2.1. The first option would be to record the IR detector signal in response to true scenes presented to the missile optics. With no available facilities³ whereby complex scenes may be artificially generated for the missile optics, and with the IR detector signal being very small, it was decided to generate input signals through a software model of the optics. This flexible method allows processing of complex scenes containing the IR signatures of targets within complex environments with relative ease.

3.2.2.1 Scene generation

Various typical missile flight scenes may be generated from the missile's point of view using an IR battlefield simulator at Aerotek. Aircraft such as the C130 transport and the Oryx helicopter were previously photographed in the air from all possible vantage points, using an infrared spectrometer in the 2 to 5 μm IR range. These pictures were then melded together in a radiometrically accurate 3D wire-frame model. This can be viewed from any aspect or distance to model an accurate IR signature from an approaching missile's point of view. The battlefield simulator makes use of these wire frames, together with topographical maps of terrain, humidity models, cloud models, surface foliage representations and geographical positions. This is done at certain times of day on specific days of the year in order to model almost any type of battlefield scenario as an IR environment.

A week was spent at Aerotek in April 1998 generating appropriate scenes from which detector signals might be derived. Four series of 10 second missile flights were created; each consisting

³ Aerotek CSIR possess facilities whereby point source targets are generated artificially in a controlled environment presented to a missile, but point source targets might not prove sufficient. It was decided to make use of an accurate software model because of its ease of implementation and flexibility.

of 1000 frames taken at 10ms⁴ intervals at 512x512 pixel resolution in the 3 to 5 μm IR range. Each recorded pixel value represents an instantaneous radiance measurement at a direct line of sight view angle between missile and target with 32 bit accuracy. This resulted in a total of 4 Gb of recorded data, which is converted to ASCII information⁵ totalling 13 Gb. An example of a scene from series 1 is provided in Figure 4.



Figure 4 C130 IR image with vertical clutter

The first three series cover straight-lined proportional navigation approaches to constant velocity C130 transports, from missile launch to target impact. Extreme vertical terrain clutter was included in the first series, since the straight lined edges of the reticle spokes in Figure 1 are known to correlate well with straight vertical and horizontal edges. The effects of such clutter on the missile's tracking ability can provide useful SID data when eventually presented to the missile in the form of a detector signal. The C130 is also useful due to its warm 4 propeller engines. Depending on the approach vector only some of these are visible from the missile's vantage point at any particular time. This represents the tracking of various point sources at the same time, an informative experiment.

⁴ Minimum possible step time of simulator.

⁵ The simulator runs on a SUN SPARC workstation and saves its output in a non-standard format that is only useful to programs written by the developers of the simulator. The only option currently provided is to convert it to ASCII, which is rather cumbersome.

Series 2 and 3 cover the C130 from different angles of approach with some horizontal clutter included in series 2 and no clutter included in series 3. The cardinal difference between series 1 and 2 is that the former represents tracking across clutter at different background IR levels, whereas the latter represents tracking along constant clutter.

Series 4 is a panning lower rear view aspect of an Oryx helicopter at 400m constant distance. The Oryx's two engines are spaced close together and should prove a less complicated target to the missile. Wire frame models are only available for the C130 and the Oryx, the main reason why only these specific aircraft were chosen for the experiment.

3.2.2.2 The software model

The software model which processes the scenes into an IR detector signal is written in Matlab, due to the relative ease with which the scenes may be handled by Matlab's Image Processing Toolbox, together with the strength and ease of use of its matrix processing capabilities. Unfortunately Matlab does, however, possess serious processing speed limitations. The following paragraphs contain a basic description of the program, which is listed in Appendix B.

The program fits a reticle mask over each chronologically presented scene, while at the same time rotating the reticle at an estimated body roll rate (the reticle is fixed to the body). The reticle centre is then moved over the scene along a circular path. The total instantaneous IR power at the detector is determined by adding together the values of all the pixels fitting through the transparent parts of the reticle mask at any given time, and multiplying the result by the area of collection. An enlarged picture of the reticle, together with measurements provided of the total look angle covered by it, provides all the information necessary to create the mask.

Each 512x512 resolution scene represents 6 degrees field of view (FOV) from the missile to the target. The size of the reticle mask was taken to be 256x256, since the reticle must fit within its specified FOV. The radius of the circular path along which the reticle centre is traced is provided by the offset angle of the front-most mirror of the Cassegrain assembly, with the conical scan starting with no offset from the centre of the look angle towards the target, i.e. the radius of the circular scan path is referenced from the centre of the scene. This implies that 106 pixels remain on each side of the scene which are not used when determining the detector

signal. The additional pixel space allows completion of reticle simulations with target offsets from the centre of the FOV of up to 1.25 degrees to any side if it becomes necessary to provide more drastic CSFM inputs to the missile when doing SID.

The program inputs include the total flight time of each simulation; the time resolution with which each instantaneous irradiance value is determined at the detector; the time steps with which the reticle is rotated; and the interval between new scenes. All four series were processed for the first 8.3 seconds of each flight, providing 33.2 seconds of IR detector signal. This is directly proportional to the radiance reaching it.⁶ After approximately 8.3s, the target's apparent size/proximity increase (in combination with the focal abilities of the optics and increased IR irradiance) may cause the TAG circuits to be activated within the missile. Since all of the following models are only suitable for far field targets of relatively small size with the assumption that TAG remains inactive, this prospect would only do harm when used in a model optimisation routine. Figure 5 illustrates a typical irradiance progression⁷ for a 10s proportional navigation approach to a target, as illustrated by processing the first image series⁸. Note the influence of two vertical clutter towers around 1s and 7s of flight. The typical Noise Equivalent Flux Density (NEFD) of the detector, around 77 °K, is between 25 and 50pW/cm². Experience has shown that the missile may track targets with an irradiance contrast of about 200pW/cm². A zoom of Figure 5 indicates that the target falls within the trackable range from the outset.

⁶ Providing that detector saturation is not reached, and keeping in mind that the detector is AC coupled to the preamplifier.

⁷ Only with DC removed.

⁸ The signal itself is around 1200 Hz, which is why only the envelope is visible.

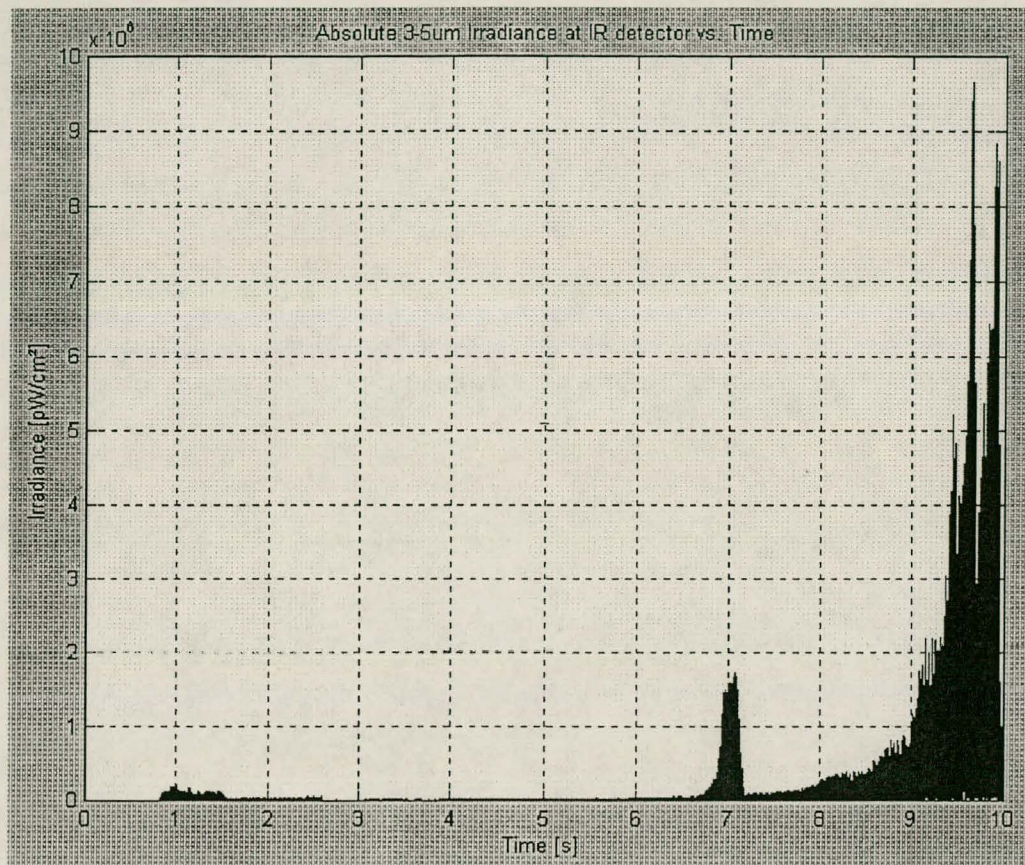


Figure 5 Typical IR irradiance envelope at detector during 10s of flight

This type of simulation requires the following characterisation data:

- Determination of the detector coupling and radiometric properties.
- Knowledge of the Cassegrain FOV and determination of the conical scan offset.
- Knowledge of the typical missile body roll rate, as well as of the gyro spin rate.
- Knowledge of the reticle form.

The specific missile analysed determines the ease/difficulty with which the parameters may be determined. Disassembly is usually required only for the first and last of these points, and most of which may be bypassed if suitable scene generation hardware is made available to replace the software.

CHAPTER 4

SIGNAL INJECTION AND RECORDING

4.1 Hardware configuration

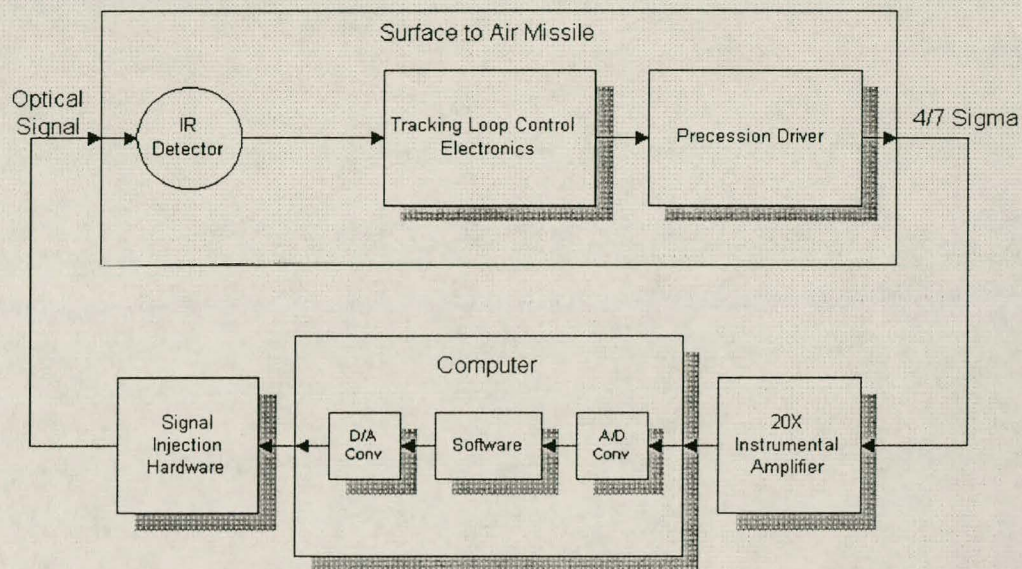


Figure 6 Hardware signal injection/recording set-up

Figure 6 provides a block diagram description of the hardware signal injection/recording set-up constructed around the missile seeker head. Aerotek, CSIR developed an optical signal injection system for IR-FM guided missiles that transforms electrical signal waveforms into optical injection waveforms that may be presented to the IR detector system. The specific details of the injection system are of a classified nature, as it is the only known system of its kind known to be in use, however, the following characteristics are known:

- When dealing with injection signal frequencies below 20 kHz the phase distortion between the input to the injection hardware and its output is negligible.
- Inputs to the injection system are saturated/limited and converted to electromagnetic light pulses at the IR detector, implying that amplitude information is lost during injection. Most of the frequency, phase and FM content of the input signal are retained. Saturation does however add additional uneven harmonics from the original input signal to the signal presented to the detector.

This type of saturated signal input is the only method available at present for use in performing signal injection on the missile. It was developed as a result of the extreme difficulty encountered when performing linear injection at the IR detector. Linear injection would require radiometrically correct infrared irradiation levels at the detector with about 20pW/cm² accuracy. Designing such sensitive injection equipment is a priority for Aerotek, but falls outside the scope of this project. The eventual results attained in this project illustrate that linear injection would result in greater model accuracy. However, in order to construct a sufficiently accurate model the saturated pulse train injection system is adequate.

Before saturation of the computer-constructed IR detector signal data may be allowed, it needs to be ascertained that the signal content is not distorted within any important frequency ranges. As saturation implies the addition of harmonics, a BPF is used to remove any signal content outside the 700Hz to 1.7kHz range surrounding carrier frequency. This is to ensure that no meaningful harmonics end up within this range after saturation. The exact form of this filter and the motivation and explanation for its use is identical to the discussion in section 5.2.2.1 and the corresponding figures. Essentially, the filtering process somewhat distorts the signal that is to be injected, but filtering remains a necessity. The response of the missile to this slightly distorted typical IR detector input provides the best known chance of activating all the possible missile target tracking modes for later modelling purposes.

Section 3.1 deals with the motivation behind the choice of the IR detector as injection input and the $\frac{4}{7}\sigma$ output as the signal to be recorded. For the purposes of this discussion the IR detector signal is considered to be a natural point of injection, as it is the missile's sensor to the outside world. The $\frac{4}{7}\sigma$ signal is regarded to be an excellent and attainable representation of the precession signal.

An instrumentation amplifier amplifies the precession output by 20 times in order to increase the A/D conversion resolution. The A/D converter within the computer has a 12 bit accuracy over a 20Vpp range, implying a minimum conversion resolution of 5mV for the least significant bit. The $\frac{4}{7}\sigma$ signal saturates at 300mVpp, implying that only 60 levels of

discretisation would be available if the signal was recorded without prior amplification. With the amplifier included in the loop, 1200 levels of discretisation over the peak-to-peak range of the $\frac{4}{7}\sigma$ signal are available.

The A/D converter is supplied with a standard 1.2kHz corner frequency low-pass anti-aliasing filter at its input. The 1.2kHz corner frequency should not affect the recorded precession signal adversely, as it is easy to illustrate that the precession signal content seldom ventures above 500Hz, even when allowing for signal saturation. The anti-aliasing filter will, however, have some effect on the recorded signal, in particular phase. It is important to keep this fact in mind when eventually using the recorded data. For this reason, Appendix C describes the construction of an accurate model of the A/D process' influence on the recording.

Section 4.2 deals with the software used to record the precession information, as well as the software used to write IR detector signal data to the injection hardware via the D/A converter.

4.2 Software configuration

Interface between the computer and the missile is provided via a 12 bit A/D and D/A converter card called an "ADAS"-card, developed at the University of Stellenbosch. The converter fits into a standard ISA slot within an Intel-based computer, and is accessible via software commands sent to the card's base address and the relevant card address offsets. The programming of the card interface/driver is not described in this thesis, but the program written in Pascal 7.0 which serves both as the PC/Converter interface and signal injection and recording system is provided in Appendix B. A basic description of the program follows:

A binary data file is read into the computer's memory. The data contains the previously saturated data that is to be sent to the injection hardware. (Note that saturation took place after the BPF was applied to the data.) The data preparation was completed using Matlab 5.11. Each injection series consisted of 83 000 data samples 100 μ s apart. The ADAS-card allows 12 bit accuracy both during A/D and D/A operation, and completes both A/D and D/A conversion using integer values to represent its input and output port amplitudes.

For example, writing 2047 to the D/A conversion port results in 10V at the D/A output, and -2048 results in -10V⁹, with each successive integer in between these values representing a 5mV change in amplitude. This allows data obtained after Matlab processing to be saved as 16 bit integers, and these integers to be written out to the ADAS-card. The D/A output representing the saturated IR detector signal is supplied as a full scale 20Vpp output saturation to the injection hardware:

These 83 000 data points correspond to 166kB of information, while only 64kB may be addressed within one memory segment in a Pascal 7.0 program running under 16 bit MS-Dos. This necessitates memory reservation and segment jumping as is evident in the program. Such segment jumps are also visible after the A/D conversion cycle, as 83 000 corresponding data words are read from the A/D port representing the missile's precession signal response. All recorded precession data is held within the computer's memory, and is only saved to a file after the injection is completed, as file reading and writing interferes with the clock interrupt regularity.

The entire injection and recording cycle is PC-interrupt driven. Interrupt 0x08H, the PC timer interrupt, is set to vector only to the injection and recording program, and its regularity is set to the closest approximation of 10kHz¹⁰ available, loading the number 119 in the 8254-type clock chip's countdown buffer.

All available data injection series are injected into the missile, and the response of the missile precession signal recorded. Section 4.3 discusses the different series injected during this phase.

4.3 Injection series I/O grouping

Four injection data series were constructed, making use of an IR battlefield simulator at Aerotek, and an optical simulation program listed in Appendix B. Seven additional series were constructed artificially to serve as additional injection material with which the missile may be characterised. Some basic signal types were constructed for the sake of redundancy and model validation, and motivation is provided as to why these specific series were constructed.

⁹ The same integer value to voltage scaling holds for the A/D converter.

¹⁰ The true resulting clock rate is 10 026 Hz.

Series Name:	Description:
Series s1	C130 Hercules at 2000m alt. 25° rear aspect proportional approach, starting 3km behind, with vertical clutter provided by “towers” in the image background.
Series s2	C130 Hercules at 2000m alt. 15° rear aspect proportional approach, starting 3.5km behind, with constant horizontal clutter provided by false horizon 0.5 degrees below target.
Series s3	C130 Hercules at 2000m alt. 15° rear aspect proportional approach, starting 3.5km behind, with only standard atmospheric distortion included.
Series s4	Oryx helicopter rear aspect view, panning from port to starboard in a circular fashion from 25° to port perpendicular to 25° to starboard perpendicular in a semi-circular fashion at 400m radius.
Series a1	1200Hz carrier with 100Hz FM at 0.1 modulation index.
Series a2	1200Hz carrier with 70, 90, 95, 98, 100, 102, 105, 110 and 140Hz FM at 0.1 modulation index.
Series a3	Same as series a2, only 0.2 modulation index.
Series a4	Same as series a1, only carriers at 1200, 1250 and 1300Hz all with 100Hz Fm.
Series a5	1200Hz carrier with wide frequency FM, 100Hz at modulation index of 1.
Series a6	White noise, Gaussian distribution with 0 mean and variance of 1.
Series a7	Variable amplitude signal modulated onto 1200Hz carrier. Modulated signal = $\sin(2\pi t)\sin(2\pi 100t)$, with 0.1 modulation index.

Table 1 Missile IR detector injection series

Series s1 to s4 contain the battlefield-simulated data for signal injection discussed in section 3.2. Series a1 to a7 were created using Matlab and are henceforth known as the “artificial” series. These artificial series range from simple FM at the known 1200Hz system carrier frequency, to complex multiple FM inputs and white noise injection. Series such as a2 and a3 provide multiple FM inputs in order to capture the missile’s response to FM input surrounding its known/postulated FM carrier frequency. Series a4 examines the missile’s response to carrier frequencies surrounding the known/postulated 1200Hz value. Series s5 provides a wide frequency span FM input closely related to the true series s1 to s4 representations, yet fully controlled and without noise. Series s6 provides for white noise system driving, a technique frequently used during SID, which could prove helpful at a later stage. Series s7 provides an amplitude modulated modulation signal which represents a model of the gyroscope LOS scanning to and fro over the target twice every second, as is used in Appendix A to model the gyroscope.

From the series and their recorded responses, two groups of data are formed: Training data and Test/Validation data. These groups are known by the names usually attached to neural network training and test phases, as they will be used for neural network training and validation. But the question remains as to which data series should be divided into which group? Some typical criteria for efficient training and test sets are set out below:

- Training data should cover the widest possible range of eventual neural network inputs. This includes frequency coverage, amplitude coverage, phase coverage, different input pattern combinations, etc. The aim is to control the network's response to as many different situation/scenario types as possible, necessitating as many different types of response examples provided during training as possible.
- Training sets should focus on system responses to typical input signals, if typical input signals exist. Neural networks are usually trained to exhibit a minimum error energy signal. This implies that the eventual relative trained accuracy of the network over a given type of data set is somewhat proportional to the amount of training data acquired from those types of data sets, when compared to the total number of data sets. For example if the IR detector usually receives input frequencies between 700Hz and 1.7kHz, then most of the training data should result from IR injection within this frequency range to assure that the network provides an accurate model of the network within this input range.
- Training sets should not be "too large." Many texts supply basic heuristics regarding the relation between neural network sizes and the sizes of their training sets. If a training set is too large, a network may encounter problems converging to error minima during training, or could make impractical time requirements for convergence.
- Test sets should include data that is unrelated to any input series with which the network was trained. This is done to judge the so-called "generalising ability" of the network.
- Test sets should include data related to the network training data as a means to confirm the network's response to typical inputs and to test the network's noise rejection capabilities.
- The neural network needs to represent the original system up to its boundary conditions, while still allowing accurate results at mean conditions and retaining fresh sets of data

which are able to test the network's response to both these conditions. A training compromise should be found between these.

Unfortunately, the above mentioned criteria may be interpreted in many ways, and given these criteria and the same series as in Table 1, very few individuals would group these series in the same way. The training set size criteria is ignored for now, being left to the actual training phase before being decided upon. If training set size appears to be becoming a problem, equal percentage cuts of each of the series within in a set will be made. The following series groupings are made:

Training Set	Test Set	Additional
Series s1, s3, a1, a3, a5	Series s2, s4, a2, a4, a7	Series a6

Table 2 Injection series distribution amongst training and test sets

It was decided to reserve half of the data series for test, and half for training purposes. Most test set to training set size distributions are about 30:70, but it was decided to place more focus on accurate test and validation of the network, before hardware implementation of the system is to be allowed.

Half of the battlefield simulator's series are grouped in each set. Series s1 represents C130 approach in a cluttered environment for training, while series s2 tests the network's response to a different cluttered environment. Series s2 also provides a counterpoint for series s3's training results, as s2 is identical to s3 excepting the added background clutter (noise). Series s4, the Oryx rear view, differs from the other three scene generated series in that the Oryx has two main IR sources in comparison with the C130's four sources. With series s4 being quite different from series s1 to s3, it was decided to choose s4 as the unrelated data input used to test "generalisation".

Series a1 represents the most basic operation of the missile, and is therefore chosen to be the central typical response required during training. Series a1 is also known as the mean condition between the boundary conditions. Series a5 is elected to cover the training boundary

conditions, since it contains the highest FM index injected into the missile, and therefore also the widest frequency span. Series a3 provides an artificial training input which covers many superimposed frequency components within the normal missile tracking loop band of operation, and contains a slightly higher modulation index than series a1 and a2. Series a2 is included in the test set as it is related to series a1 in terms of its modulation index but not its content, and related to series a3 in terms of content but not modulation index. Series a2 covers part of the “related test signal” criteria for the artificial series. Series a7 differs from the other artificial series, because of its AM content within the modulated signal. It is included in the test set as a “generalisation” test amongst the artificial series.

The remaining series a6 is not classified as part of either the test or the training series, as yet. Because of its white noise content, one signal window from this series has no relation to any other signal window. If necessary different parts of it can therefore be used to supplement both training and validation.

CHAPTER 5

RELEVANT MODELLING TECHNIQUES

5.1 Gain vs. Compromise

Before continuing, it should be made clear that there are very specific compromises to be made when venturing towards rapid modelling of the SAM. The majority of these compromises hold for most other types of control system models as well. Table 3 provides a concise overview of deterministic reverse engineering SID (System Identification) and SM (System Modelling), in comparison with rapid non-deterministic modelling.

Table 3 Deterministic modelling vs. Rapid modelling

Criteria	Deterministic Modelling	Rapid Modelling
<i>Time Requirements</i>	A slow system, as every subsystem of the missile is to be disassembled and analysed to be able to construct accurate models. (Approx. 14 to 24 months)	Very little SAM disassembly required, allowing fast modelling. (Approx. 3 to 4 months)
<i>Accuracy</i>	Most accurate method, next to receiving a comprehensive model from the manufacturer	Accuracy determined by various factors such as: processing techniques, modelling structure and topology, optimisation routines, amount of available data etc.
<i>Characterisation</i>	A true characterisation method where the exact workings of the SAM are to be determined and understood.	Limited information about the SAM is revealed, depending on the exact type of model and SID signals used for I/O signal response.
<i>Manpower</i>	Wide field of expertise coupled with intimate electronic knowledge required to be able to disassemble the SAM, not to mention understanding the functions of each subsystem. Various specialists may be necessary.	General knowledge of SAMs required, i.e. some previous experience in disassembling SAMs. More rigorous signal processing required. Requires 1 or 2 suitable engineers.
<i>Simulation time</i>	Accurate models are usually non-real-time, but approximate models may be real-time depending on available computing power.	Intended for both real-time HIL simulation and non-real-time modelling. Structural and optimisation compromises may be made to decrease simulation time.
<i>Client Market</i>	Exploitation engineering, countermeasure development, flight simulator training, total threat analysis.	Rapid countermeasure development and testing. Rapid missile tracking capability analysis.

Clearly the aim is to get the most accurate model in the least amount of time. That would imply that only a limited number of features should be extracted from the missile, and that these features should be easily extractable in order to save as much time as possible. This thesis looks specifically into the possibility of using artificial neural systems when attempting to minimise the number of features to be extracted, while at the same time analysing the accuracy of the resulting models. Each type of rapid modelling approach has its strengths and weaknesses. The following paragraphs analyse some of the modelling approaches attempted throughout this project, in keeping with Table 3. The initial approaches require almost no knowledge about the missile tracking loop (as in 5.2.1), progressing to those which require some structural knowledge of the system in favour of model accuracy (such as 5.2.2 and 5.2.3). During the course of this study, many different model structures were fitted to the available data, but only the best results of each approach are illustrated. Illustration of all attempts would be monotonous, space-consuming and of little additional value. However, a comprehensive analysis of the eventually implemented approach is provided.

5.2 Rapid modelling approaches attempted

5.2.1 Solo Network

The most basic approach to modelling the tracking loop with an artificial neural based system is the construction of the entire model using only a neural network (NN). Even before attempting this approach, it is clear that its structural difference in comparison with the true control system might quite easily be problematic. It should, however, be attempted for the following reasons:

- If effective, this approach would definitely be the fastest rapid modelling technique available, and therefore must be considered.
- With the network uninfluenced by surrounding structures, this would be an excellent test platform to determine what types of networks are suited to the temporal signal injection and rapid modelling problems.
- This is a gateway to determining the type of training functions that could converge to the required output waveforms.

The network architecture, training algorithm, activation functions, size and IO-data format must be decided on. These features cannot necessarily be uncoupled from each other, but they each contain certain criteria that may be decided on separately.

5.2.1.1 Architecture

The tracking loop is an Infinite Impulse Response (IIR)¹¹ system, and therefore contains various temporal transients. These transients are distorted by certain non-linearities that inevitably lurk within the system. Not many networks are suited to temporal signal processing, significantly narrowing the search. The three well-defined possibilities are:

- Elman networks; consisting of three neurone layers (input, hidden and output layers) with the hidden layer copied into a fourth so-called “context” layer and fed back to the hidden layer during the next input sequence. This is an example of a feedback network. All feedbacks and forward feeds are weighted.
- Jordan networks are similar to Elman networks, except that the output layer is placed in context instead of the hidden layer.
- Multi-layer perceptrons with history; generally consisting of no more than 3 or 4 layers of neurones, with the first and last layers being the input and output layers respectively. The network itself contains no feedback, but the input layer is provided with the past history of inputs by a pre-processing algorithm. An example of this would be a five neurone input network, of which the five inputs are the last five samples of a temporal input signal. One sample is shifted out of this input “register” at every time step, while the latest sample is simultaneously shifted in at the opposite end of the register. This is a Finite Impulse Response (FIR) approximation of a system, consisting only of multiple zeros which approximate pole behaviour.

Various other (lesser known) architectures do exist, but they are essentially combinations of those three outlined above. Examples of these include Elman-Jordan networks, Elman-History, fully recurrent networks and others. Jordan networks are unfortunately not especially suited for Single Input Single Output (SISO) systems, because of their limited feedback capabilities when only placing one output value in context.

¹¹ Apparent from the continuous feedback control structure.

Elman networks are well-suited for SISO system models, but tend to have problems representing long time constant behaviour with their context layers, since they need to be trained for both stability and accuracy. When training, they cannot necessarily uncouple their feedback poles, resulting in some poles being pulled uncontrollably into instability when slow poles (near the complex plane origin) are formed. Inevitably, a lower pole limit cannot be passed, as instability cannot be tolerated.

This has serious repercussions for systems with greatly varying input and output signal frequencies, such as the missile target-tracking loop¹². The context layer struggles to maintain enough of the input layer history from the high frequency signal, to be able to form the low frequency output. It eventually attempts to fit a high frequency signal with a low frequency AM envelope to the output. This phenomenon is well known to individuals with Elman network experience, but is not widely published.

The final architectural option, the multi-layer perceptron with history, is the only network providing any satisfactory results, as its modelling flexibility and abilities are directly linked to the structure and amount of history provided to the input layer.

5.2.1.2 Training algorithms

Neural network design, training, simulation and validation are completed within the Stuttgart Neural Network Simulator (SNNS) software package, running under Linux. This software was chosen because of its cost-free availability, extreme flexibility in neural network design and the fact that it is able to convert a trained neural network into C-code.

Anyone who has made use of neural systems most likely invented their own network structure and training algorithm as well. SNNS provides more than 30 different types of training algorithms, including a myriad of variants of well-known algorithms such as backpropagation. Completing a detailed analysis of the merits of each of these would be a study all by itself. It was decided, however, to apply as many of the algorithms as possible, as it would be sheer folly to disregard a possible solution simply because it makes use of a lesser-known training algorithm. The results were not surprising. Only four of the algorithms produced even limited

¹² Input at approx. 1.2 to 1.3 kHz, Output around 100 Hz.

convergence to the required output signal: Standard Backpropagation, Backpropagation with Momentum, Scaled Conjugate Gradient (SCG) training and Resilient Propagation (RPROP).

Backpropagation makes use of the output error size of the network, combined with the relative weights that produced the error, to determine the amount of feed-forward weight adjusting to be done. The last two methods are in essence algorithms which determine the slope of the output error surface, and then tune the network weights and biases to steer the instantaneous error along a likely maximum downward slope towards a probable minimum error or trough. The SCG approach delivers good results, both in terms of convergence speed and accuracy. RPROP fares only slightly better than SCG, making use of a slightly customised form of SCG, whereby the inter-neurone weight changes (Δw_{ij}) are determined by :

$$\begin{aligned} \Delta w_{ij} = & -\Delta_{ij}(t-1)\eta^- & \text{if } S(t)S(t-1) < 0 \\ & -\Delta_{ij}(t-1)\eta^+ & \text{if } S(t) > 0 \text{ and } S(t-1) > 0 \\ & \Delta_{ij}(t-1)\eta^+ & \text{if } S(t) < 0 \text{ and } S(t-1) < 0 \\ & 0 & \text{else} \end{aligned} \quad \text{Equation 1}$$

where $S(t) = \frac{\partial E(t)}{\partial w_{ij}}$, E = The network's output error and $0 < \eta^- < 1 < \eta^+$

η^- = decreasing factor, specifying the factor by which the update-value Δ_{ij} is to be decreased when minimising the network's output error. Typical value = 0.5.

η^+ = increasing factor, specifying the factor by which the update-value Δ_{ij} is to be increased when minimising the network's output error. Typical value = 1.2.

Whenever there is mention of networks being trained throughout this thesis, the training was completed using SCG and/or RPROP algorithms¹³.

Figure 7 clearly illustrates the performance of the four different algorithms that were useful. Each of these graphs represent the best results obtained with each of the algorithms for a 300x15x1¹⁴ multi-layer perceptron structure (as described in 5.2.1.1), trained with

¹³ All 4 of the converging algorithms were applied during most training sessions, just to be sure.

¹⁴ See 5.2.1.4 for network size considerations.

approximately 10 000 input patterns. The SCG method never reaches the minimum RMS error that RPROP attains; but following 3000 training epochs, both RMS curves flatten out with only about 0.02 RMS of a difference. The backpropagation algorithms both flatten out fairly early, yielding instructive, yet unsatisfactory, results.

From Equation 1, it is seen that RPROP is to SCG much like Backpropagation with Momentum is to Backpropagation. This is also evident from Figure 7, where the initial partners in these “pairs” converge faster than their respective associates. In sum, both RPROP and SCG algorithms clearly perform the best, reaching the minimum RMS error values within a “reasonable” number of epochs.

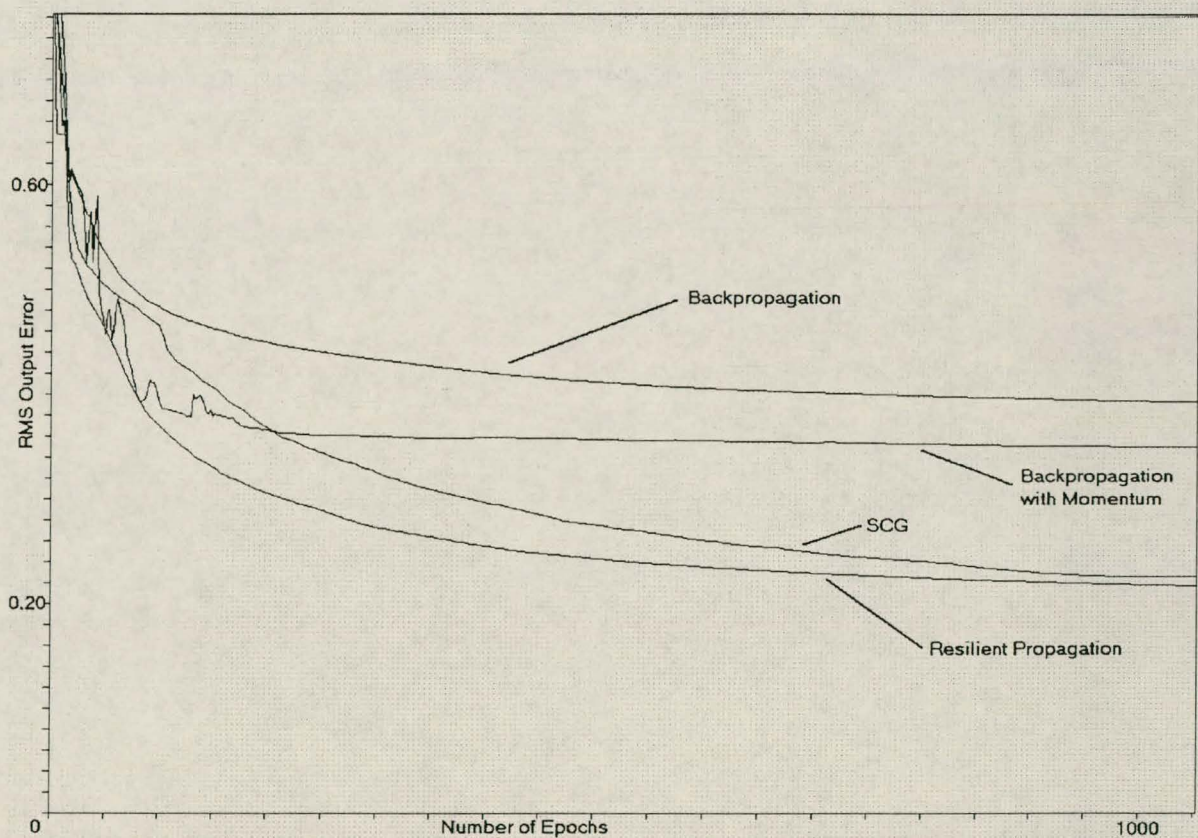


Figure 7 Typical RMS error progression during training using four different algorithms

5.2.1.3 Activation and output functions

Activation functions determine the response of each separate neurone to the sum of its input signals. Output functions are sometimes added following the activation function outputs in order to scale or format certain outputs. Activation functions are generally difficult to decide on, but there are certain simple guidelines to be followed for this tracking model:

- Bipolar activation functions are suitable for problems where zero-mean input and output signals are to be followed/generated, as they allow positive and negative outputs, usually scaled between -1 and 1 .
- It would be wise to assist the network with signal saturation ability, either by using a limited output function or by making use of “steep” activation functions. (The required precession signal (the network output) is often saturated.)
- Typically only differentiable and continuous activation function should be used when fitting a neural model to a continuous signal, especially when applying gradient approaches to network training.

Some well-known suitable activation functions include:

$$\text{Bipolar Sigmoid Function:} \quad f(x) = \frac{2}{1 + \exp(-\lambda x)} - 1 \quad \text{Equation 2}$$

λ = Steepness factor¹⁵, x = Sum of neurone inputs

$$\text{Bipolar Hyperbolic Tangent:} \quad f(x) = \tanh(x/2) \quad \text{Equation 3}$$

When $\lambda = 1$ we find that Equation 2 = Equation 3, with this case illustrated by the solid line in Figure 8. The “steepness” of the function may be increased by increasing λ .

Natural saturation occurs in the true tracking system’s output signal because of precession amplifier saturation when high gyro precession rates are required. A simple aid for the network would be to turn the output function of the sole network output neurone into a limiter, with unity gain and saturation at the required signal output levels. This relieves much of the pressure on the network to be able to model saturation effects (i.e. having to add extra frequency harmonics to its output) and it allows the default SNNS setting of $\lambda = 1$ to be used whilst using the sigmoid activation function. The other neurone output functions are simply chosen as unity gain, so as not to obscure the network’s training operation by any unnecessarily rigid boundaries.

¹⁵ In SNNS $\lambda = 1$ is the standard setting.

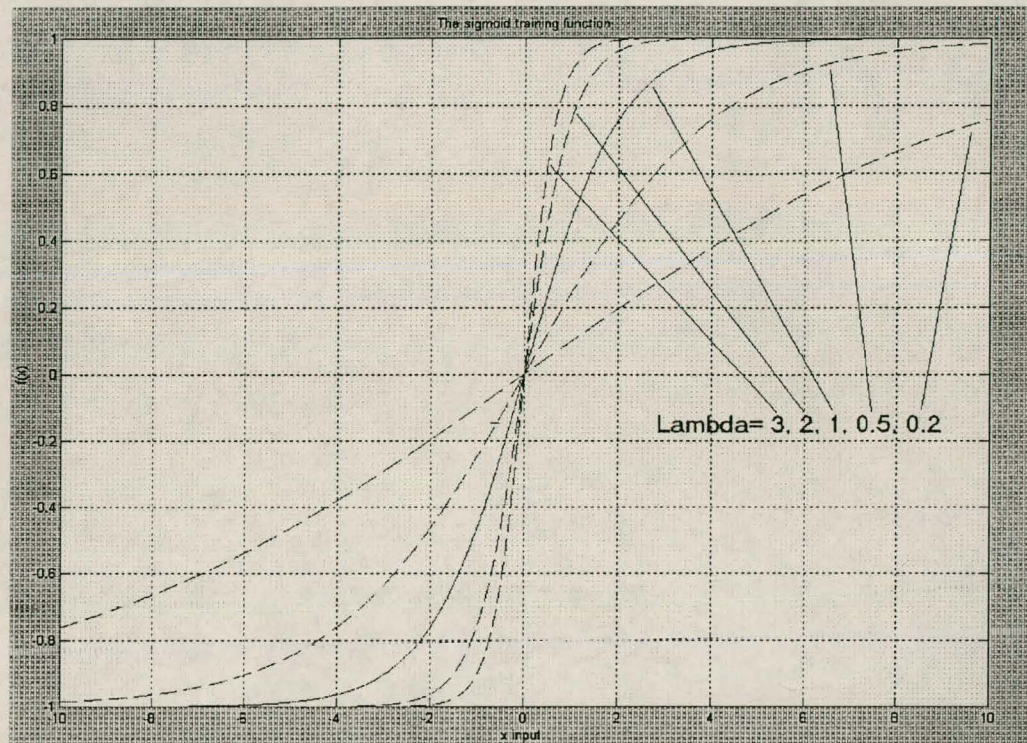


Figure 8 The sigmoid training function

5.2.1.4 Network size

Various heuristics attempt to find a correlation between network size, accuracy (i.e. the number of neurone layers and the number of neurones within each layer) and efficiency. But how would an “effective network”, in this instance, be judged?

- An effective network needs to converge to a sufficiently small RMS error during training. There might quite probably be a smallest possible error for any given network architecture that is not sufficiently small.
- It should exhibit a sufficiently small error when tested with inputs that were not provided as training data, but do represent the same characteristics as the input data.
- It should be able to generalise with sufficient accuracy (i.e. have some ability to process inputs that fall outside its scope of training). Smaller networks are usually considered to be better suited for generalisation than large networks.
- It should be small enough for a desktop computer to determine each new output value in real time, i.e. at 10kHz.¹⁶

¹⁶ Even after peripheral overheads such as A/D and D/A conversion are included, in order to be suited for HIL.

Most of the size heuristics deal with factors pertaining to the separation of input patterns into classes by making use of hyper-planes. A network should be able to construct enough hyper-planes within a pattern space to be able to classify each input to fit within a certain cubicle. For instance, some texts define the number of necessary hidden neurones within a 3 layer network, J , to be $J = \log_2(M)$. Here, M is the number of pattern classes we seek to separate in a linearly separable problem, while accepting that J is much smaller than the dimension of the input space. Unfortunately, very few texts supply suitable heuristics when dealing with temporal signals, where there are no definite classes in which the outputs are likely to fall. Under these circumstances, it is usually evident that multi-layer perceptron networks provide more satisfactory results, the larger they get. This can be related to the fact that the network apparently attempts to classify its instantaneous output into as many pattern classes (M) as possible, in order to give a more enhanced approximation of a continuous signal. What some texts fail to mention, however, is that larger networks also require more training patterns in order to train effectively, as it is futile to train a network capable of representing x number of different patterns while only presenting it with x minus y number of examples.

There are many practical issues such as processing time, pattern separability and signal frequencies that quickly overshadow many of the heuristics, especially when dealing with temporal signals. Even after an indication has been given as to the necessary size of a network, one must still experiment with different sizes and choose a network generating the best results. Some additional considerations arising from the experimental approach are outlined below:

- The input signal is sampled at 10kHz and contains valuable FM information within the 1kHz to 2kHz frequency band. The output signal mostly contains only important frequency components within the 100Hz to 200Hz range. Even from the first training session it was evident that enough time sample history of the input signal is to be provided to the network to cover at least 1 or 2 periods of the output signal. The longer the time history the longer the network takes to train, but the more accurate its output becomes. A good compromise between network size and accuracy seems to result when 300 input samples (lasting three output periods at 100Hz) are provided as history. Very little increase in efficiency results when the network input is enlarged. Decreasing the input to 200 samples adds approximately 30% to the RMS output error for training

data, increasing from about 0.22 MSE on a 0.84 RMS training signal, to an MSE of 0.29.

- The input signal described above could not have provided any respectable results without a hidden layer of neurones. With every extra hidden neurone connected to the input layer, we now add 300 extra feed forward connections! Some experiments regarding the size of the hidden layer seem to indicate that 15 to 20 hidden neurones are required to provide the best results. The RMS errors given above were derived from a 300x15x1 network.
- Even before we analyse the accuracy of our results, it is evident that no desktop computer would be able to process a 4515 connection network such as this in real time at 10kHz. The C-code provided by SNNS completes a processing task in about 60s on an Intel PII 300MHz based computer, that should last 10s in real time.

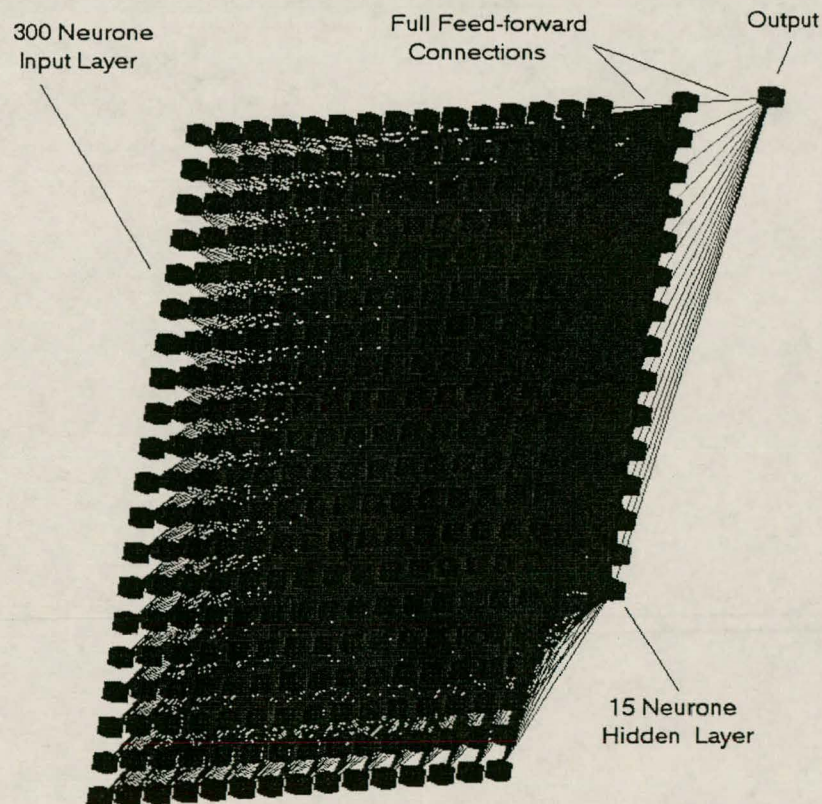


Figure 9 A 300x15x1 Multi-layer Perceptron Network

5.2.1.5 IO-data formats

Very little pre-processing is completed with this type of model architecture. The input signal is placed in a matrix consisting of as many rows as there are input patterns, and the input patterns are generated by placing the past 300 input signal samples in a row vector. For training purposes SNNS then requires the corresponding output layer values to be placed next to each of the rows. All input samples are scaled between -1 and 1 to provide a uniform and limited input to the network. This poses no problems because the input values are mostly saturated FM. Output signals contain natural saturation because of precession driver clipping. This saturation level is simply scaled down to between -1 and 1 as well, to provide a uniform network output. Section 5.2.1.3 deals with the output function of the sole output neurone of the network. This is taken to be a limiter with unity gain and saturation below -1 and above 1 .

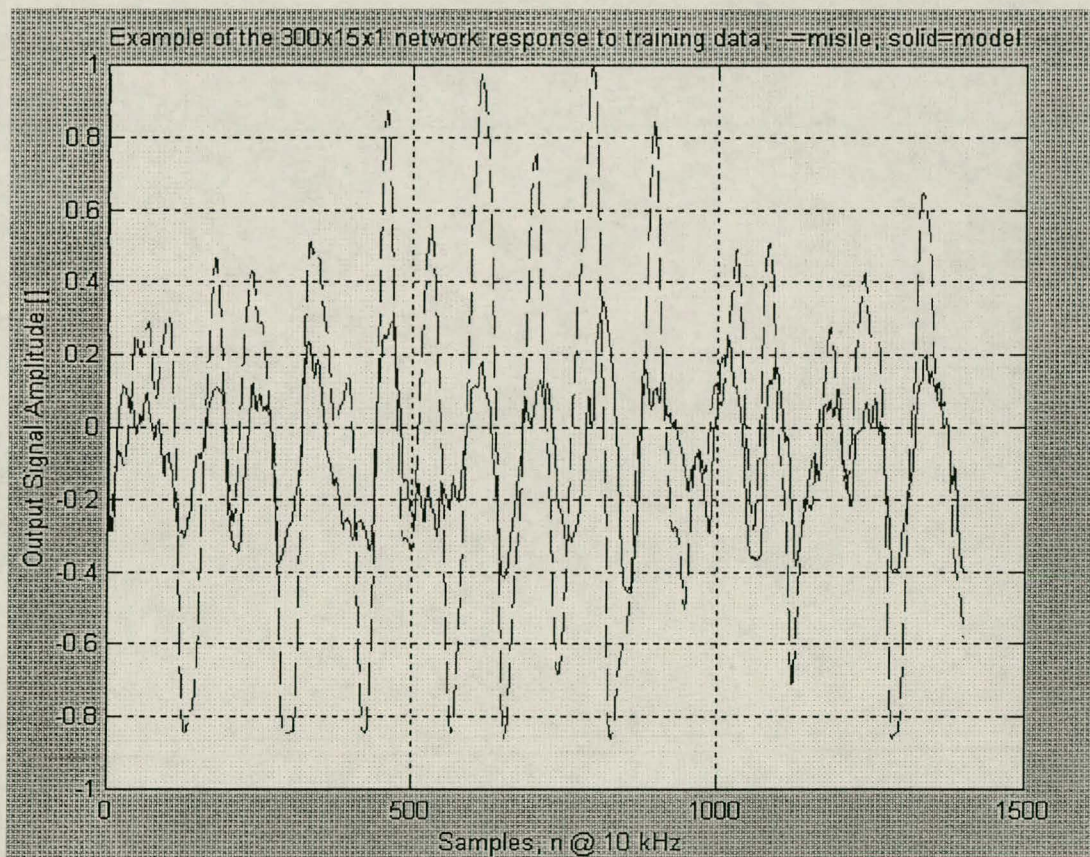


Figure 10 Network output response to training data

Looking at Figure 10, a typical output of the best trained 300x15x1 network that was constructed can be seen, compared to the true missile's output. This is clearly not adequate! The amplitude differences are unacceptable and some phase shift can be seen (e.g. at

approximately 800 samples on the x-axis). Even the network's response to test data should provide more accurate results than Figure 10 before it could be accepted as adequate.

Even with all of the arguments posed throughout the previous paragraphs, there is no guarantee that an accurate network can be constructed. The only supposition that is made is that the most accurate network model possible will be constructed when attempting to model the entire tracking loop with nothing more than a single neural network.

What proves of greater import than the results of this network (which we knew at the outset to be a risky possibility for success) is the lessons learned about the types of networks and training scenarios which best suite this type of problem. The following paragraphs deal with the refinement of the structures developed through this approach, until an accurate neural based model is constructed.

5.2.2 Generic pre-processing with sub-sampling to reduced network

The previous discussion resulted in a network with two primary problems:

- The network output is inaccurate when compared to the missile's true precession signal.
- The network is too large to be processed in real-time.

In order to deal with these problems further analysis of missile's internal functions and processing techniques is needed, without necessitating any further disassembly of the missile seeker. At this point it is necessary to make a certain general assumption about the missile:

Assume that the missile makes use of a demodulation technique to retrieve and decode the information encoded onto the IR detector signal, and that all the necessary information about the demodulation process may be obtained by simply looking at the reticle pattern in Figure 1 and its interaction with the optics.

This type of assumption may seem far-fetched at first, but if a list is made of the information obtained by looking only at the reticle and the optics, the validity of the assumption is clarified:

- Even a cursory inspection of the optics quickly shows that the missile makes use of CSFM¹⁷ to encode target directional and line of sight information onto the IR detector signal. Some clear give-aways are the symmetrical, radial spokes on the reticle and the angled front mirror on the Cassegrain assembly.
- Figure 1 clearly shows that there are twelve main spokes on the reticle and in Chapter 4 it was found that the gyro spin rate is 100Hz. This implies that the carrier of this FM signal lies around 1200Hz (12 x 100), depending on the body roll rate during flight.
- With the gyro spin frequency known, and the gyro precession coils twisted directly around the magnetised gyro-optics, it is known that the precession signal functions in the range of 100Hz. This is also in keeping with typical CSFM, as the gyro spin rate usually makes up the bulk of the signal FM-ed onto the detector signal carrier.

Not all IR guided missiles make use of such easily analysable optics, but much of this information can be extracted from many SAMs and even AAMs. Examples that spring to mind are the Russian made SAM 7, SAM14, SAM16 and AA7, and several Chinese versions of these missiles.

5.2.2.1 Pre-processing

It is now clear that the network in 5.2.1 must be able to demodulate the incoming signal and filter out all the resulting unwanted frequency components. At the same time, the special characteristics of this specific missile's demodulation process must be mimicked, and any other additional unknown processes characteristics within its structure must be captured. A tall order indeed! An attempt will therefore be made to make use of a general/generic demodulation system in order to demodulate and filter the IR detector signal before providing it to the network. There are certain advantages and disadvantages to this type of pre-processing. These are outlined below:

Advantages:

- The network's task list is drastically reduced, allowing it to focus on imitating the unknown characteristics of the missile.

¹⁷ Conical Scan Frequency Modulation

- The output signal of the demodulation process is known to operate mostly within the 100 to 200Hz frequency range. There is little need to sample such a signal at 10kHz. The low frequency output of the pre-processing therefore allows down-sampling to 1000Hz. This is five times the necessary sampling rate needed to remain in keeping with the Nyquist principle for 100Hz. Also, assurance is needed that there are no high-frequency components within the signal that may be mixed down/mapped into the 0Hz to 1kHz bands by the down-sampling process. Figure 11 is an illustration of the typical frequency content of the recorded precession signal. From this figure it can be ascertained that a sufficiently small amount of information is carried on the signal above 500Hz, especially in relation to the typical signal peaks surrounding 100Hz. Saturation within the true precession signal causes the small peak still visible at 500Hz, but with pre-processing being the new source of the signal fed to the network the signal is not simply limited, but a limiter is added to the network's eventual output instead.

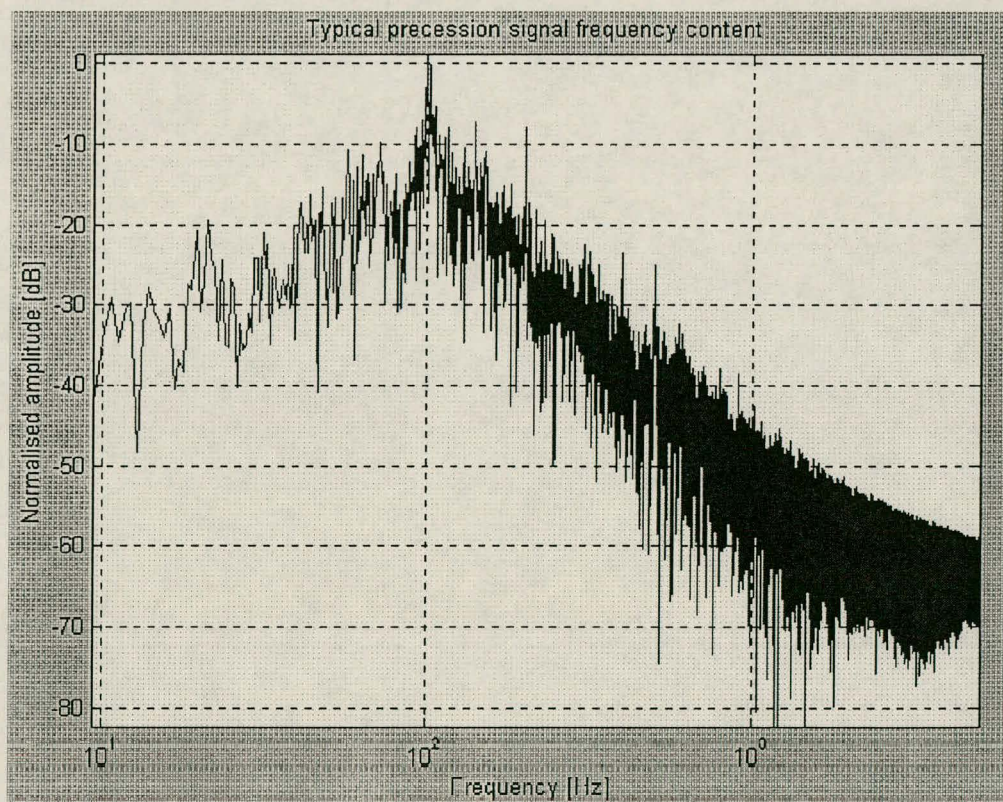


Figure 11 Typical precession signal frequency content

Down-sampling to 1kHz allows the necessary retention of only 30 samples within a history vector supplied to a multi-layer perceptron. This enables the presentation of three periods of the typical 100Hz precession signal to the network¹⁸.

This enables the reduction of the necessary network input layer size from 300 neurones to no more than 30 neurones! If the network provides sufficiently accurate results, the network needs processing only at 1kHz, instead of the previous 10kHz. A major step forward indeed!

Disadvantages:

- The missile tracking loop is known to be non-linear, implying that not all of the demodulation characteristics bypassed by the pre-processed demodulation may be retrieved by adding a network after the process itself (i.e. commutativity does not hold, but is sure to hold partially, maybe even largely, depending on the specific signal dealt with).
- Certain frequencies and other signal information that may be removed or distorted during pre-processing might be important for the network to be able to train properly.

Both of these possible disadvantages have to be kept in mind when designing the pre-processing system, but cannot be eliminated with any guarantee.

A typical pre-processing system consists of an input filter and amplifier (preamplifier), a demodulation system and an output filter which removes any remaining unwanted frequencies (esp. those caused by demodulation). These generic modules now have to be constructed.

The pre-amplifier is designed to be a bandpass filter with its passband being the frequency range surrounding the FM input carrier at 1.2kHz. The signals generated by the reticle/optics simulations in 3.2.2.2 provide insight into the necessary bandwidth (BW) of the preamplifier, but cannot possibly provide any certainty about the true missile pre-amplifier's BW. Frequency-phase relationships between the input and output of the preamplifier are unknown, with the best estimate centering around constructing an amplifier with as little phase distortion around 1.2kHz as possible. It is clear that certain phase and amplitude relationships will have to be corrected by the neural network, but it would be wise to design all the pre-processing networks to warrant as little signal correction as possible.

¹⁸ Refer to 5.2.1.4.

Figure 12 illustrates the frequency response of the 4th order elliptical filter that finally provided the best results after network training was accomplished. The filter was constructed using Matlab's signal processing toolbox. A 4th order elliptical filter was used because of its zero degree phase shift at centre frequency, and relative ease of implementation.¹⁹ The 3dB BW lies between 800Hz and 1.7kHz, with the -50dB stop bands starting below 629Hz and above 2064Hz. This filter has 4.25 degrees of phase lag at 1.2kHz, with a 3dB passband ripple.

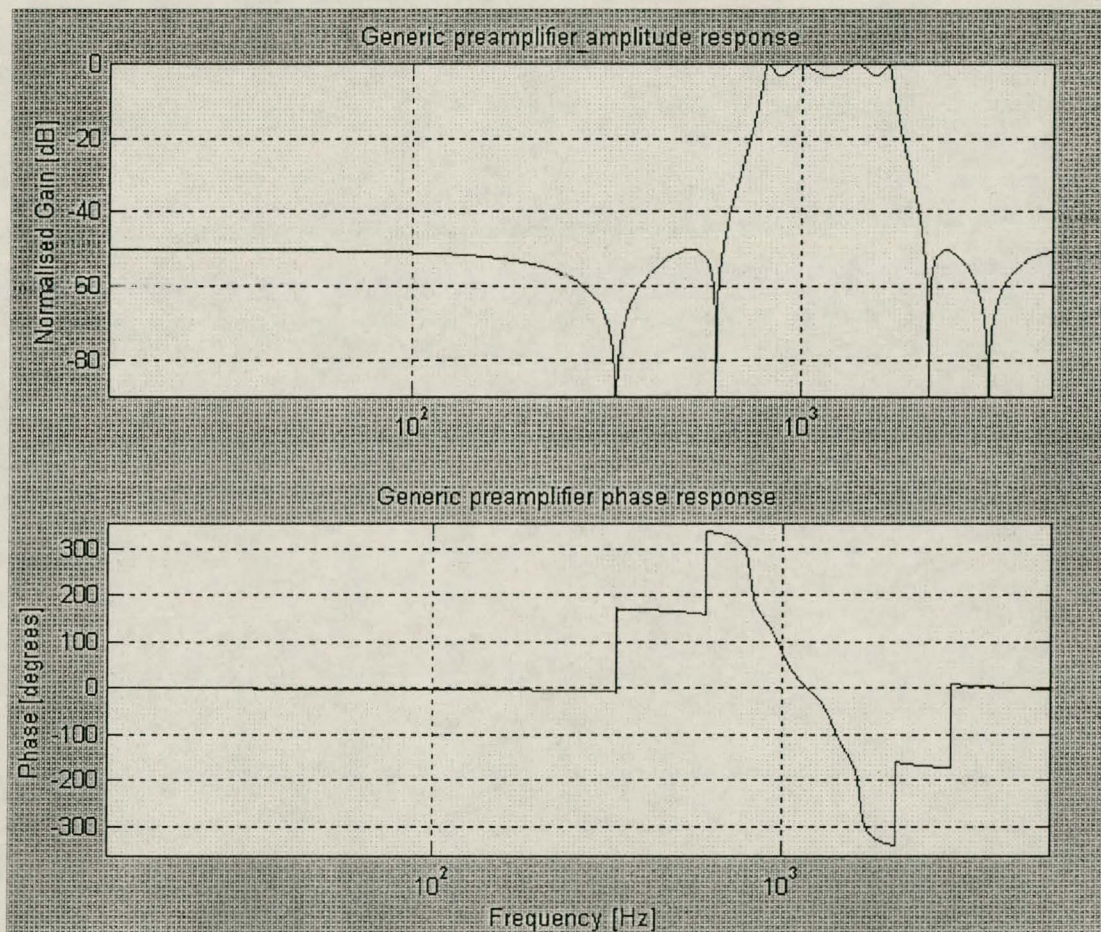


Figure 12 Generic preamplifier frequency response

A FM demodulator typically consists of a signal amplitude limiter, a differentiator followed by an AM discriminator (such as a peak detector) and a lowpass filter (LPF). FM signals become AM signals when differentiated, and the LPF removes the high frequency mirrors resulting from demodulation. The initial amplitude limiter is typical of FM demodulation, where AM-type automatic gain control is replaced by pre-filtering the input signal to fall within a specified

¹⁹ Unlike FIR Kaiser-Window and Least-Square filters with linear phase relationships but high differential orders. Also note that Matlab's "sptool" function determines the necessary filter order.

frequency band. The signal is then saturated, which adds uneven harmonics to each of the frequencies within the signal. Each harmonic is scaled in amplitude by $\frac{1}{n}$ from its originating frequency component, where n = harmonic number. It is also worthwhile to note that any of these harmonics that may be above 5kHz (the Nyquist folding frequency) will be ambiguously reflected back below 5kHz, due to the symmetric frequency domain property of sampled signals. For example, a 7kHz harmonic in a signal sampled at 10kHz will be indistinguishable from a 3kHz signal; and a FFT of such a signal will indicate signal content at both 3kHz and 7kHz. For the purposes of the following demodulation approach, the assumption will be made that only the original frequency component and its 3rd harmonic is to fall outside of the demodulation band (i.e. 700Hz to 1.7 kHz). 5th harmonics and above are deemed of little importance due to the scaling of at least a fifth of their originating frequency amplitude. A good demodulator would not make such assumptions, but little choice is left, because of the 10kHz sampling rate. The preamplifier rejected all frequencies above 2064Hz by at least 50dB, which implies that no high frequencies exist that will interfere within the 700Hz to 1.7kHz range (without being at least a 5th harmonic). Low frequency signals below 629Hz have also been rejected, ensuring that no meaningful low frequency harmonics can end up between 700Hz and 1.7kHz. (e.g. $629 \times 3 = 1.887 \text{ kHz}$, with extreme preamplifier rejection even beforehand! Only the 15th harmonic of this frequency folds back beyond 5kHz into the demodulation bandwidth.) Figure 13 illustrates how the positions of some of these harmonics may be visualised. Note the reduction in frequency component amplitude with increase in the harmonic number.

Ideal differentiators have a frequency response that is linearly proportional to frequency, given by $H_d(w) = jw$ with $-\pi \leq w \leq \pi$ where w represents the number of radians per sample for a discrete signal. The corresponding unit sample response of $H_d(w)$ is:

$$h_d(n) = \frac{\cos(\pi n)}{n} \quad \text{with } -\infty < n < \infty, n \neq 0 \quad \text{Equation 4}$$

High order FIR filters are often used to approximate Equation 4; but with the sampling rate of the input signal being about 100 times faster than the expected demodulated signal's frequency it was decided to simply subtract the values of successive input signal samples from each other

in order to provide a linear approximation of the slope of this signal for every time step. This method adds considerable noise at high frequencies relative to the sampling rate, but the noise will eventually be filtered out when isolating the required demodulated signal from the differentiator output, through filtering.

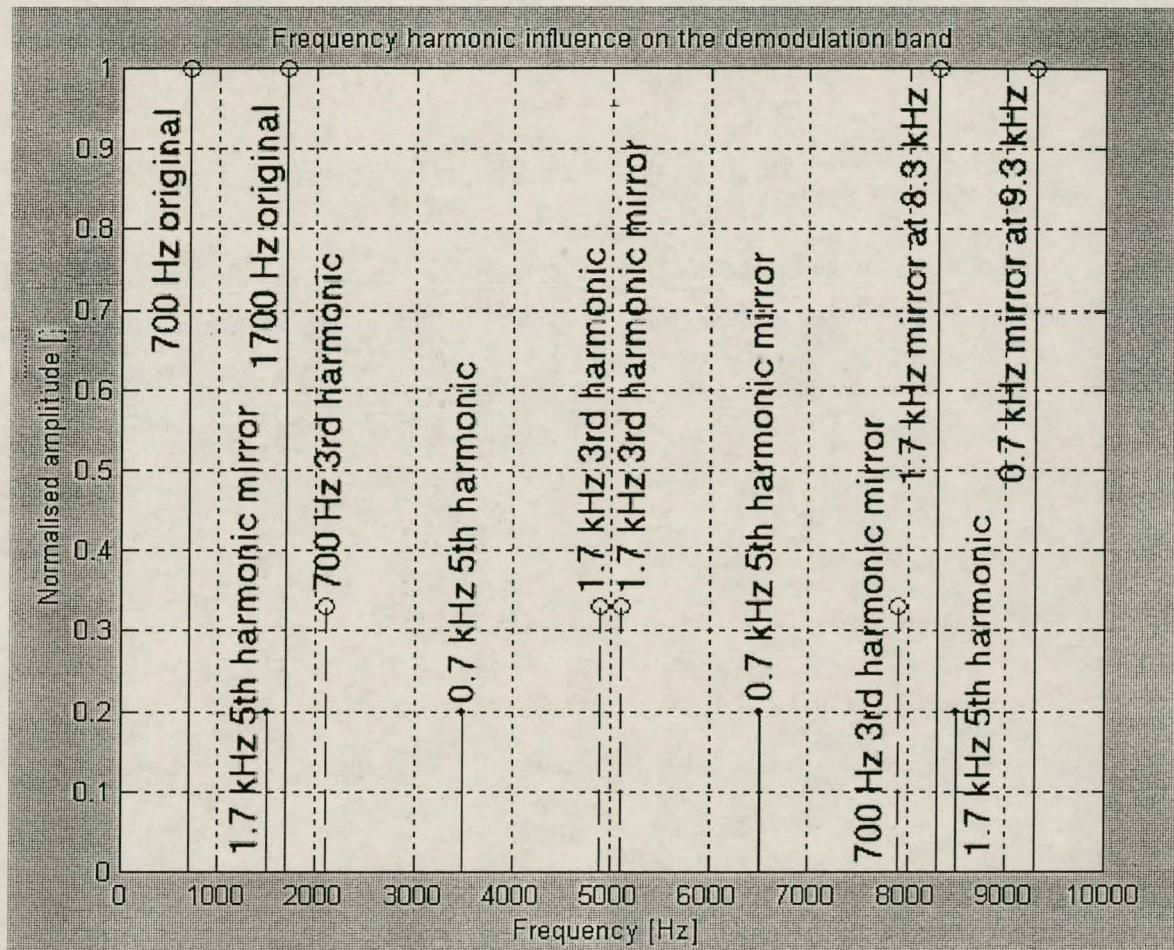


Figure 13 Harmonic distribution because of filtering and saturation

The absolute value of the output of the differentiator is determined as pre-emption to envelope detection, which is the next logical step during demodulation.

Finally, the demodulator's output needs filtering in order to remove unwanted high frequency components, including the added noise and inter-modulation frequencies, with the filter acting as envelope detector. Analysis of the recorded missile precession signal indicates that the output contains frequencies mostly between 80 and 300Hz, when output saturation is minimal.

This is the only reflection available regarding the output filtering stage leading up to the precession signal. The following filter, called a precession amplifier, was designed to assure similar frequency representation in the pre-processing output signal:

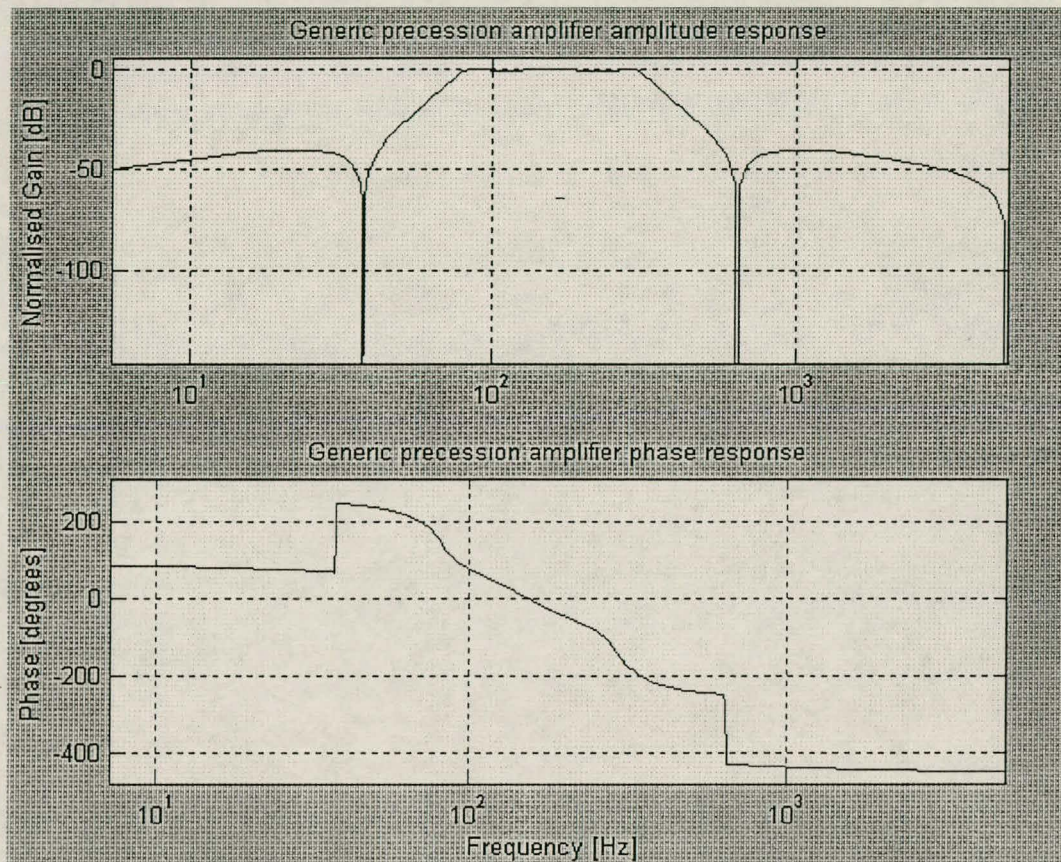


Figure 14 Generic precession amplifier frequency response

In Figure 14 we note the 3rd order elliptical filter's 40dB signal rejection of frequencies below 40Hz and above 571Hz. The passband lies between 80Hz and 300Hz, with a 1dB passband ripple. The passband, as prescribed by the recorded signal frequency content, does not allow a 0 degree phase shift at 100Hz, leaving 74 degrees of phase lead at 100Hz with the filter in Figure 14. An elliptical bandpass filter was once more used because of its small phase delays at passband frequencies, compared to for example, Chebyshev and Butterworth structures.

Figure 15 provides an example of a typical network training data window after pre-processing, in comparison with the recorded missile output over the same period. This example is illustrated after down-sampling to 1kHz. Note the apparent similarities such as relative

frequency content and signal amplitudes. It is now up to the network to complete the final fit of the pre-processing data onto the recorded data.

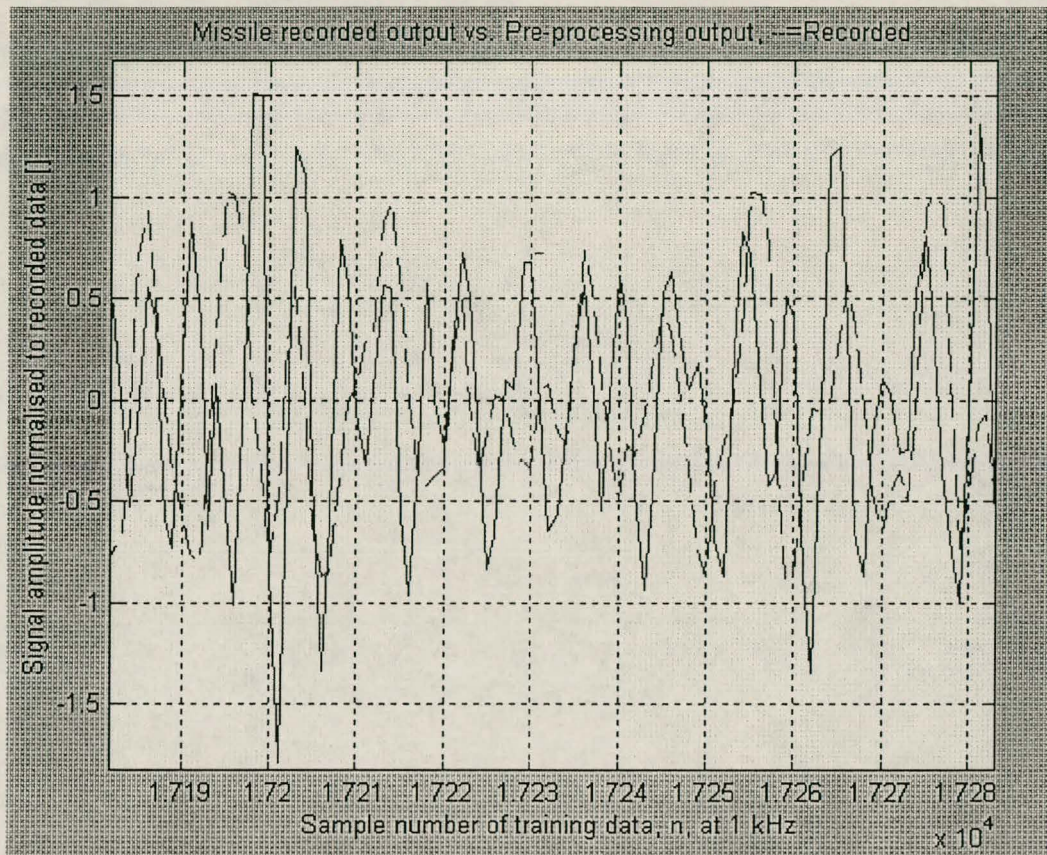


Figure 15 Typical pre-processing output vs. recorded missile output

5.2.2.2 Network issues and results

With pre-processing completed simulated precession signals are now available which are to be fitted onto recorded precession signals. The input/training signals provided to the neural network have also been down-sampled from 10kHz to 1kHz. The true recorded precession signal is not down-sampled as such, as only one sample of this signal is presented to the network for every vector of 30 down-sampled input samples. 30 input samples at a 1kHz sampling rate correspond to the same window of data that was provided as input to the network in 5.2.1. The training methods are also completed in the same manner, and the best results were obtained with the same number of hidden neurones (i.e. 15). The following bulleted list illustrates some sampling rate issues requiring attention:

- Although a 1kHz sampling rate is able to represent the necessary frequency content of the precession signal, the zero-order hold circuit on the output of the D/A converter that will eventually need to control the missile's gyroscope would cause severe phase distortion.
- Down-sampling a pre-recorded precession signal, that contains most of its data around 100Hz, from 10kHz to 1kHz could limit the signal form provided to a network especially when only short signal intervals are used for training. (E.g. a fixed amplitude 100Hz sinusoidal signal sampled at 1kHz would repeat itself every 10 samples. The same signal with varying amplitude, when first sampled at its sinusoidal peak, would be transformed into a signal where every tenth sample corresponds to the 100Hz signal's peak.) This type of pattern will not occur for long periods of real-time implementation when the input signal is not sampled with any synchronisation pertaining to itself, and because the gyroscope does not spin at exactly 100Hz (i.e. relative signal phase drift occurs.) Training a network with such a fixed signal form could distort its function as it easily focuses on such an input pattern. When implemented however, it will not be presented with such a pattern, causing poor quality output/modelling results.
- The network is trained by providing only one output exemplar for every 30 input samples. If the assumption is made that the network's output errs with a fixed, symmetric variance around the desired output value (for a certain trained accuracy), it will be possible to increase the accuracy of the average output estimate. This can be done by shifting the input samples along in 0.1ms (10kHz) steps, but presented 1ms (1kHz) apart. (i.e. signal sampling and network operation still occur at 10kHz, but only every 10th sample is supplied to the network.) The gyroscope itself would base its control around the average estimate of such a network output as it acts as a lowpass filter because of its inherent damping capabilities.
- Detector signal sampling occurs at 10kHz because of the high frequency content of this signal. A network supplied with new input vectors at 1kHz would then need to run at a different clock rate when the eventual structure is realised as a computer program.

It is found that keeping the signal sampling and network operation rates at 10kHz, while presenting 30 samples (ten samples apart) to the network input, solves all three of these problems; while at the same time reducing the required processing time for the network to about a tenth of that experienced in 5.2.1. The simulation in 5.2.1.4 that lasted 60s in order to process 10s of data, should therefore now last about 6s, which allows movement into the

domain of real-time processing. Practical tests during which the network is called 100 000 times (which represents 10s of processing) indicate that the network performs exactly as predicted.

The advantages of this system include:

- Zero-order hold phase shift at precession frequencies, which is greatly reduced because of the higher sampling rate.
- The 10kHz single-sample shift operation assures that the network is trained to be familiar with various forms of input signal windows, with peaks and troughs occurring throughout the input signal vector at various instances in time.
- The effective average output signal estimate is clearer and more accurate because of the added output samples and the LPF action of the gyroscope.
- All the aspects of the modelling process now occur at the same frequency, which simplifies the eventual software encoding of the model.

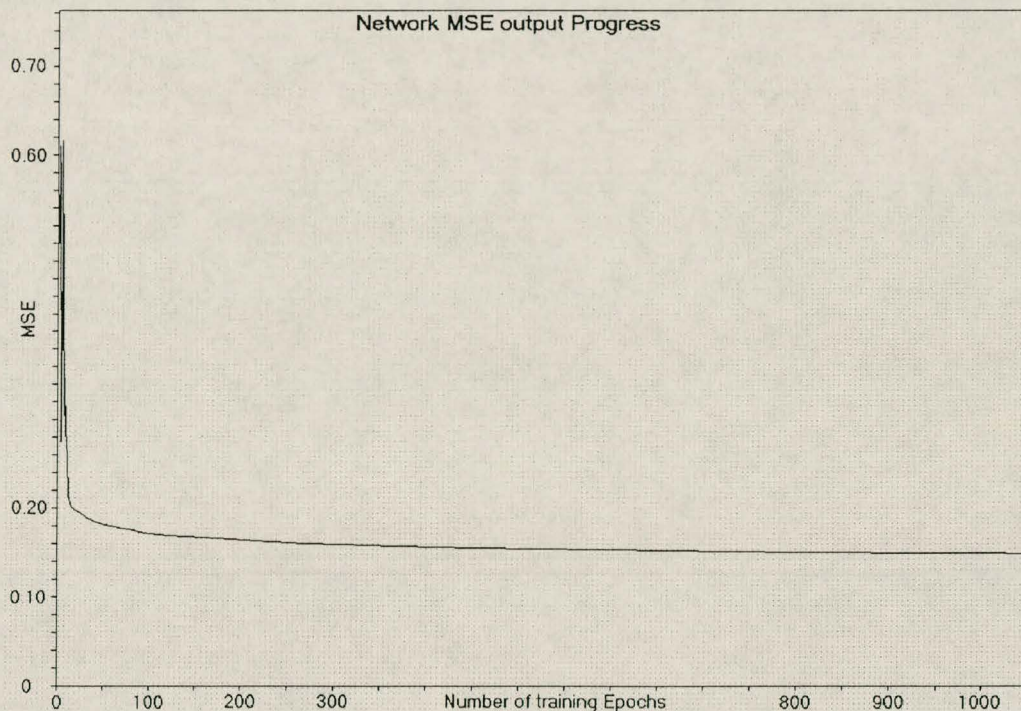


Figure 16 MSE training progress of neural network with generic pre-processing

Figure 16 illustrates the response of the neural network during training²⁰. After 1000 training epochs the network MSE evens out at about 0.15. This translates into an average error of 0.387 per output sample with the training signal having a RMS amplitude of 0.603. Figure 17 provides an illustration of the time domain results of the system. The upper half is a window over the sample area where the best model output is attained. Clearly there is very little difference between the model and the true missile response. This is, however, a response to a very simple IR detector input signal, where 100Hz is FM-ed onto a 1200kHz carrier with no noise.

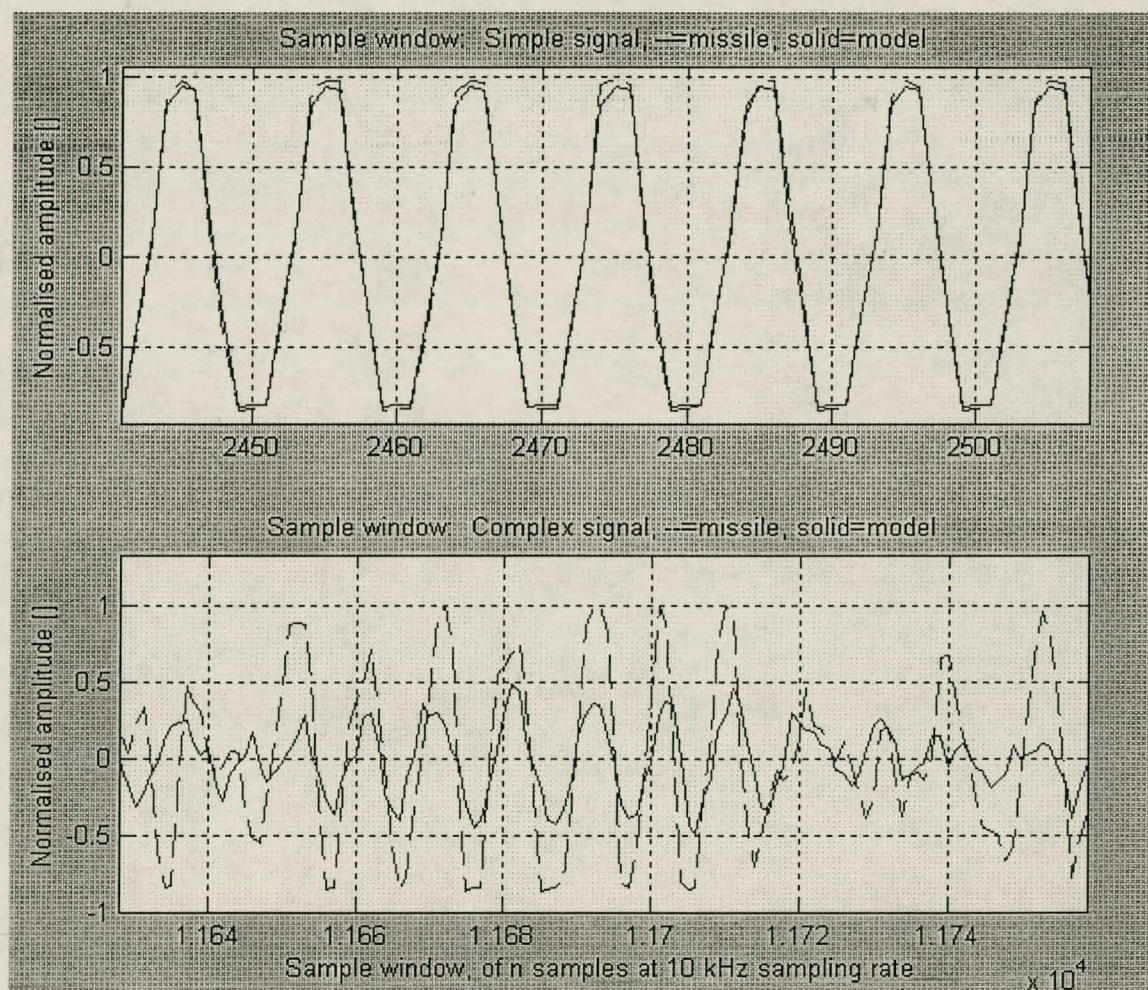


Figure 17 Response of network with generic pre-processing to training data windows

The lower half of Figure 17 deals with the response to a more realistic IR detector input signal, being an example of one of the signal series derived in 3.2.2.2, specifically a C130 rear aspect

²⁰ This was found to be the network that delivered the best overall results in section 5.2.2.

approach. The network does not model the missile's tracking loop with sufficient accuracy when faced with this type of input. Even test data should provide more accurate results than that in the lower half of Figure 17 in order for this to be submitted as an accurate (or even adequate) model.

What is clear, however, is that the results attained are more promising than those in 5.2.1. A network that is able to perform its operation in real time at 10kHz, while at the same time being more accurate than its predecessor now exists. The MSE has been reduced from 0.22 to around 0.15. The following section expands on this to provide a more accurate modelling solution.

5.2.3 Sub-sampled, optimised structural pre-processing to a reduced network

5.2.3.1 Optimised pre-processing

5.2.3.1.1 Background

The existing model structure consists of a generic pre-processing system which demodulates the IR detector signal. This is followed by a down-sampling procedure which provides the previous 30 pre-processing samples (that are spaced 10 samples apart at any given time) as input to a neural network. The neural network is a bipolar sigmoid activation multi-layer perceptron with 30 input neurones, 15 hidden neurones and a single output. Each network layer is fully feed-forward connected to its successive layer. The existing structure is able to model the missile's response to simple input signals with sufficient accuracy, but more typical noisy/advanced input signals appear to be too complex to handle properly, even during network training. In order to know how to increase the model's accuracy, we need to identify its possible sources of weakness. Three main weaknesses are identified:

- Even after careful considerations regarding the pre-processing structures, some essential signal components may still be irreparably distorted or removed.
- Pre-processing may add signal components to the system that distort its output.
- Non-linear systems, such as the missile's tracking loop, do not support mathematical commutativity (i.e. the influence of existing structures within the missile cannot necessarily be ignored during pre-processing and then simply added afterwards by a neural network). For example, including a limiter in a demodulator where it is supposed to be is much more

desirable and simpler than attempting to add its effect after the signal has undergone rigorous processing.

The most accurate form of modelled pre-processing would be to analyse the missile electronics' structure, and then to reconstruct this within the pre-processing system. This type of approach does not fall within the scope of rapid modelling. Instead, a pre-processing structure using generic filter sections will be optimised by means of linear bisection routines in order to assure that the pre-processing system is as accurate as possible. Its output is already assumed to be an approximation of the desired precession signal. From the following paragraphs, it will be clear that the structure of the new pre-processing system differs very little from the original attempt; the only difference being the replacement of the differentiator with a steep edged notch filter common to FM demodulator systems. This is in place of the first order approximation in 5.2.2.1. The new approach resulted from the need to be able to optimise not only the pre-processing filters, but also the differentiation process.

5.2.3.1.2 Preamplifier modelling

A further alteration, which has a more profound ideological impact, is the addition of an accurate pre-amplifier model. The need for an accurate pre-amplifier model does not arise from the need for an accurate pre-processing structure, but rather from an eventual practical HIL signal injection point of view. The IR detector signal can be as small as several nano-amperes, impossibly small for injection into any average A/D converter. As the missile's pre-amplifier is easily identifiable (having only two power connections, a visible co-axial cable connection to the detector and an output stemming from it) it was decided to make use of the physical pre-amplifier itself as link between the detector signal and the A/D sampler when performing the eventual HIL simulation. As a result, a pre-amplifier module was removed from a missile and its transfer-function and signal saturation levels were characterised using a Signal Vector Analyser. The modular design of the pre-amplifier, its easily apparent wiring structure and its ease of access within the missile seeker²¹, ensured that this entire task took no longer than two hours to complete. (Experience does play a serious role when attempting disassembly of such missiles.)

²¹ Only eight screws and 4 solder joints need to be undone.

Having a very accurate model of the true pre-amplifier at this stage of modelling would certainly seem to have a positive effect in decreasing the output error of any subsequent models. Yet, it will become evident that the pre-amplifier model does not appear to be the main role-player when it comes to model accuracy, but rather the demodulation system. As an example, a non-accurate pre-amplifier system is also optimised in order to show that this modelling approach is truly more accurate than its predecessors. In reality, it would be sheer folly not to include the true pre-amplifier structure within any following optimisation and modelling structures. It is essential to include all physical components within the HIL loop inside the modelling structure, if accurate signal relationships between the computer and the outside world are to be optimised.

It should be noted, however, that the preamplifier characterisation is not necessary when constructing a fully software model, which is the eventual purpose of this model.

5.2.3.1.3 Pre-processing optimisation structure

The following pre-processing structure is now provided for optimisation; the intent being to represent the precession signal with increased accuracy even before a neural network is applied. Note that all models are initially in the s-plane (analogue signal frequency plane).

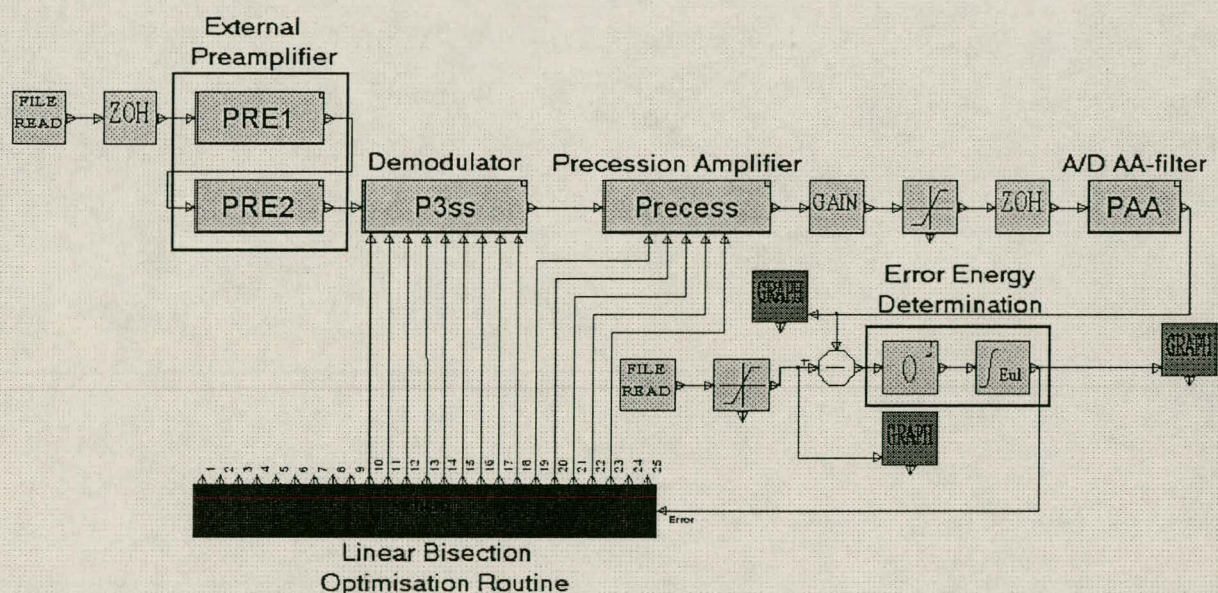


Figure 18 Pre-processing model optimisation structure

Figure 18 provides an illustration of the pre-processing model's optimisation structure. Data injected into the missile (Chapter 4) is passed through a zero-order hold (ZOH) circuit that represents the D/A electronics' ZOH, before its injection into the missile's pre-amplifier. The pre-amplifier in this figure is a two-stage implementation of the transfer function which was extracted from the true missile preamplifier. The preamplifier is followed by models of the demodulator and precession filter. Fourteen parameters will be optimised within these two sub-systems in order to assure an accurate model fit onto the recorded signal injection response data. An additional file read input block visible in the centre of Figure 18 brings the missile's recorded precession output into play.

Two limiters are visible, and provide an equalisation in saturation between the model response and the true system response. The limiter at the recorded precession data input (read from file) is redundant. A ZOH and anti-aliasing filter represents a model of the A/D circuitry through which the precession data was recorded. Appendix C deals with the derivation of this model. Finally, an error energy determination system can be seen, which serves as input to the parameter optimisation routine.

Figure 19 provides insight into the demodulator routine. The notch filter used for input signal differentiation purposes is clearly indicated. A standard 3-pole-2-zero notch filter was implemented as a first iteration initial condition in order to reduce optimisation time. The bisection routine then optimises the parameters of this structure into the appropriate values. The initial notch lies at 300Hz. The limiter at the left-hand-side input to the notch filter, and the absolute value implementation towards the right-hand-side, are explained in 5.2.2.1.

A LPF provides initial envelope detection of the signal provided by the ABS value block. Two ZOH blocks are also visible. These are included before optimisation occurs, because it is already anticipated that the s-plane system in Figure 19 will be implemented in the z-plane (discrete plane). Due to the non-linearities between the filter sections, it will be necessary to compute the z-plane transfer functions separately. As the process of transformation necessarily adds ZOH blocks before an s-plane system before computation takes place, including the ZOH blocks during optimisation assures that the transfer of these filters into the z-plane causes as little signal distortion as possible. The blocks labelled 1 through 9 represent the inputs to this module through which the internal parameters are adjusted.

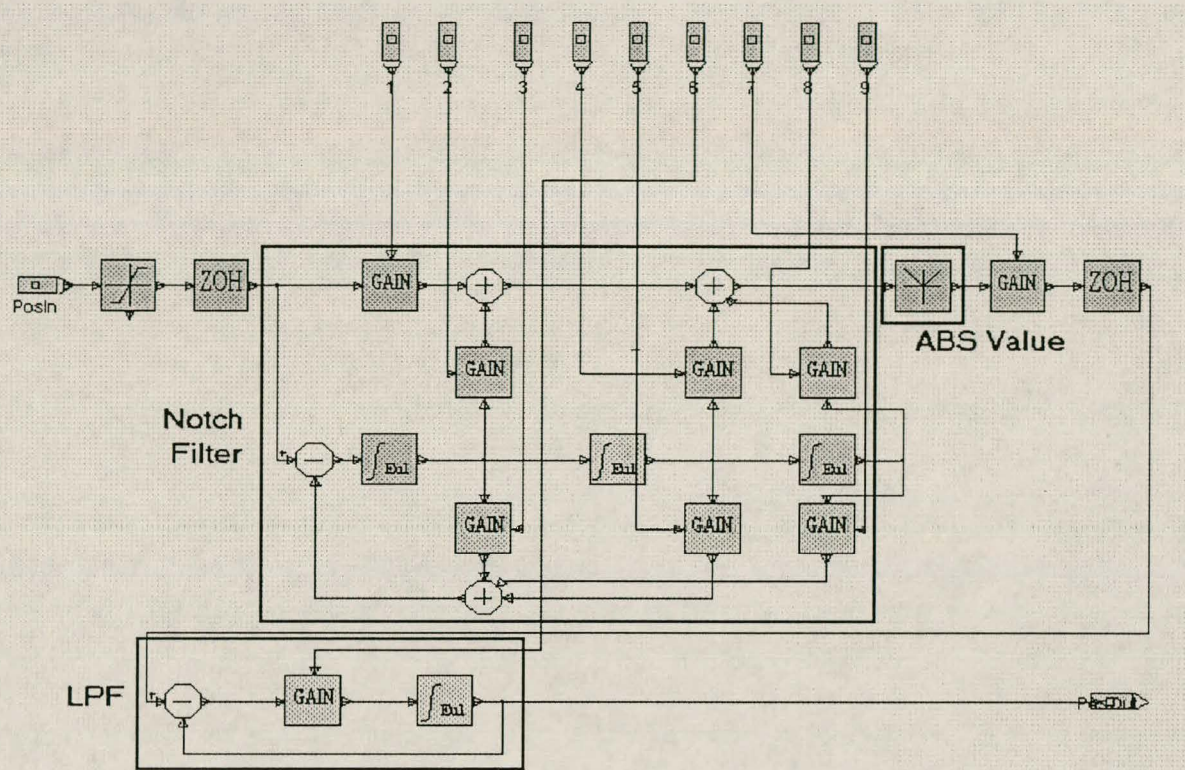


Figure 19 Pre-processing demodulator structure

The precession filter in Figure 20, described in 5.2.2.1, is the final block within Figure 18 requiring explanation. Initially, the precession filter was implemented exactly as in the initial pre-processing algorithm. After some optimisation runs it was discovered that the parameters within certain parts of the structure faded into near non-existence during optimisation, until it was evident that only the structure in Figure 20 remained. It is interesting to note that this model structure is very similar to a model structure that was deterministically derived by the author during 1997.

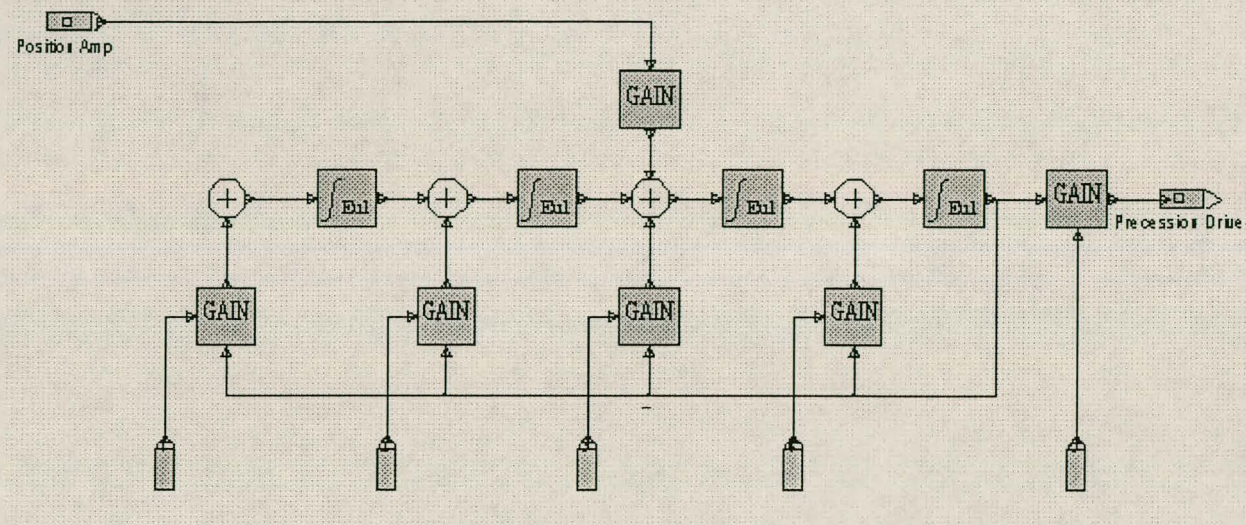


Figure 20 Pre-processing precession amplifier structure

No ZOH is evident in Figure 20, as this entire model is z-transformed together with the final LPF filter in Figure 19. No non-linearities exist in between these structures, eliminating the need for separate z-transforms with ZOHs in between.

5.2.3.1.4 Optimisation results

The error bisection algorithm is implemented as a standard function within Simuwin. The training data for the bisection procedure consists of approximately 10 seconds of data obtained from the training set in Table 2²². It required approximately three days of continuous optimisation to adjust the 14 pre-processing model parameters in response to these 80 000 pre-recorded missile I/O samples. Eventually, the MSE over the training samples evened out at 0.05. The MSE of the best system in 5.2.2 could only even out at about 0.15, even with a neural network attached! The training data series provided to the pre-processing network here in 5.2.3.1 do not coincide exactly with that processed by the system in 5.2.2, but are fairly similar. *Indications show that this new type of pre-processing delivers much improved results even before a neural network is added to the system.* Some accuracy loss is expected after the z-transforms of the pre-processing filters have been completed. True comparisons between this new approach and the previous approaches will only be undertaken after its neural network is attached and the same data is processed by this new system as was processed in 5.2.2.

²² See Chapter 4 for details.

The following figures indicate to what degree the optimised pre-processing system responded to both training and test data. 16 seconds of data from series s1 and a7 were processed by the Simuwin model, and written to files. Figure 21 illustrates the response of the system to a known window of training data, while Figure 22 illustrates equivalent responses for known test data windows, i.e. data arising from the same series as the training data but not used for optimisation purposes.

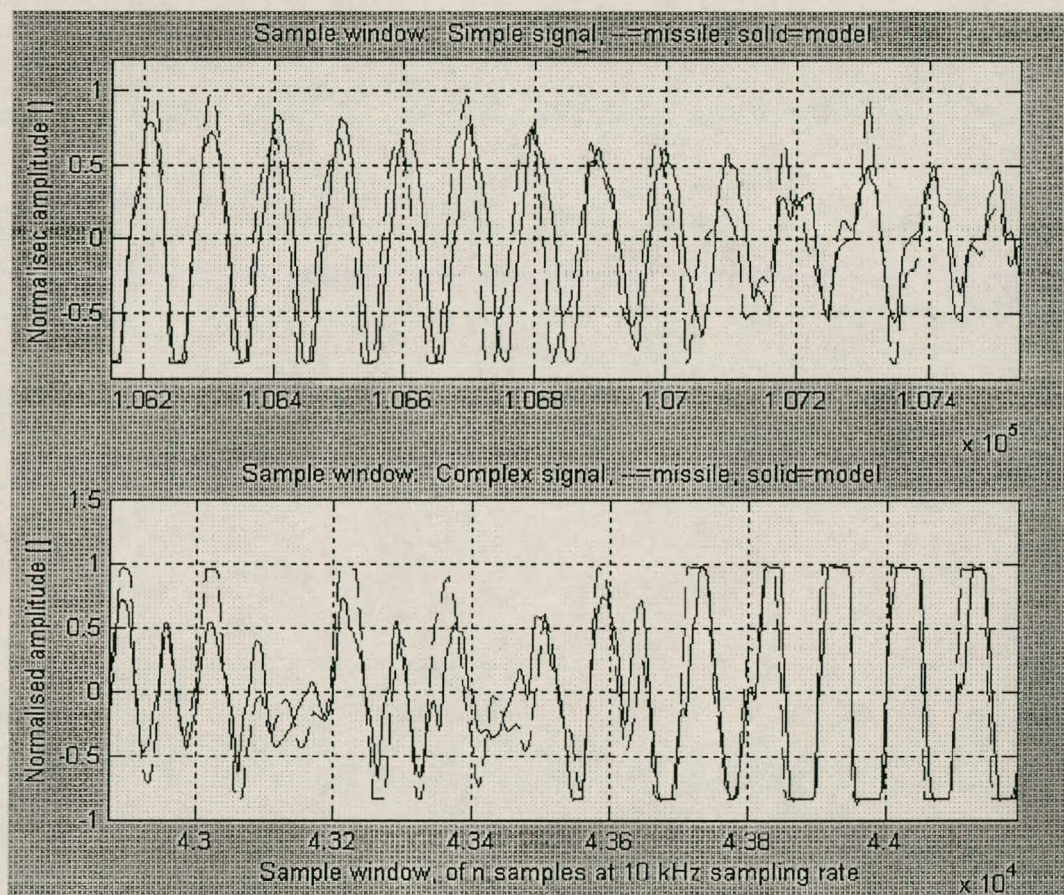


Figure 21 Typical pre-processing training data output results

In Figure 21 the upper half of the illustration deals with the response of the pre-processing system to series a7, considered to be a simpler excitation than the lower half of the figure. This section deals with the trained response to series s1, a proportional navigation approach to a C130. The model fails to deliver an exact fit for either of the output signals, but many more similarities are evident than after the initial pre-processing of 5.2.2.1 (Figure 15). There seems to be little difference in accuracy between the responses to series s1 and a7.

It must be noted that phase and amplitude relationships during periods of required saturated output are fairly accurate. This is an important part of the signal, as it is known that high gyroscopic angular acceleration rates are only attained when the precession signal is saturated. Non-saturated outputs, though important, are less so than relatively high amplitude-precession windows, as they arise from situations where the target is basically centred within the gyro line-of-sight (LOS). Computing the SSE for use during the bisection routine, aids in assuring higher percentage output accuracy for higher amplitude signals. Larger amplitude signals usually display greater output error values, and are now weighted more by the squaring process within the SSE optimisation.

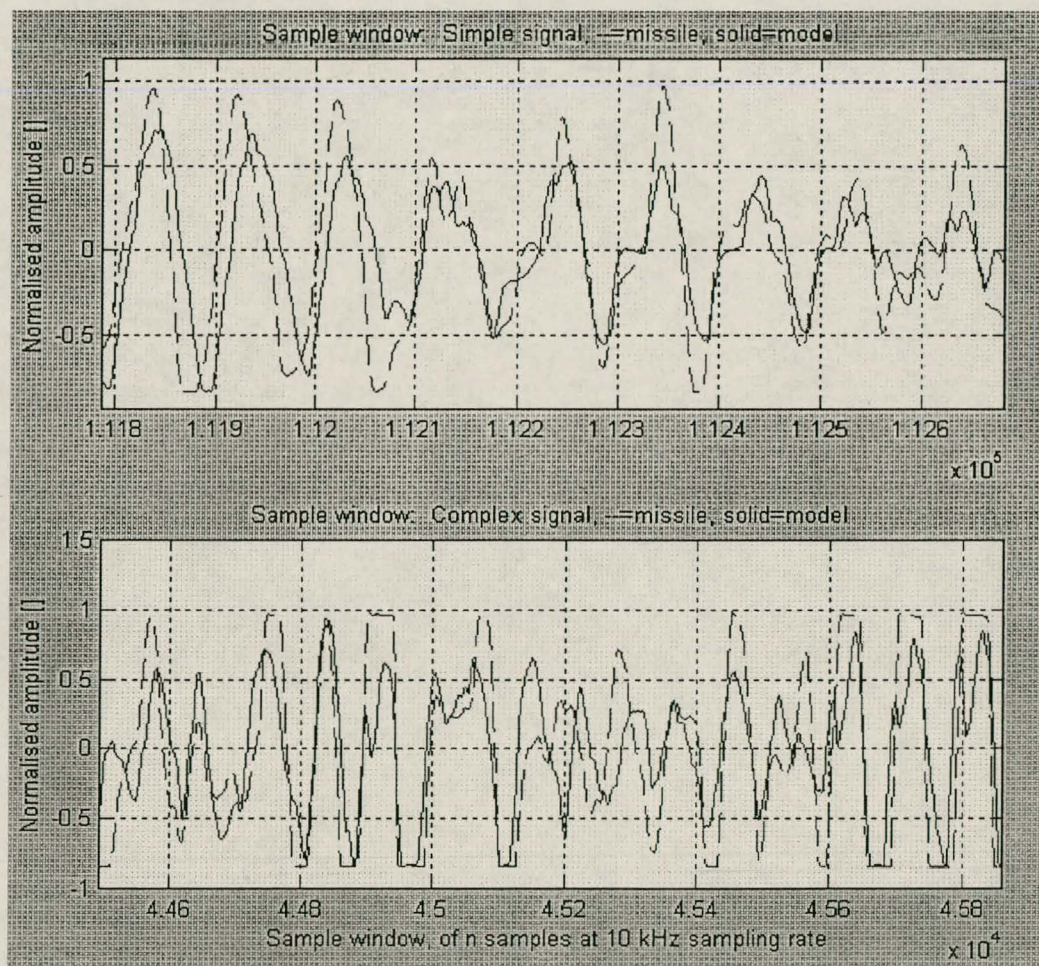


Figure 22 Typical pre-processing test data results

In Figure 22 little difference can be seen between the modelling error incurred when verifying its response to test or training data. This is typical behaviour for an optimised transfer function type model, especially when the model's architecture is closely related to the true system's

architecture. The MSE over the course of the test data model validation proved to be 0.064, compared to the 0.05 MSE incurred by the training data. This is an acceptable loss in accuracy, and is still ostensibly more accurate than even the most successful previous attempts at fitting training data.

Additionally, the optimised pre-processing system's response to a totally unknown signal series needs to be checked. In this case, the series derived from a rear-aspect view of an Oryx helicopter (series s4). Figure 23 illustrates the typical response of the system for the series s4 input. The output is not fully accurate, but the similarities are clearly visible, with a MSE of 0.07 for the 8 second series duration. Phase distortion is the most visible error, which will be left to the neural network to correct. Unfortunately, phase errors are unacceptable as they easily cause gyro nutation, having the gyro continuously circle the target instead of moving its LOS directly towards it.

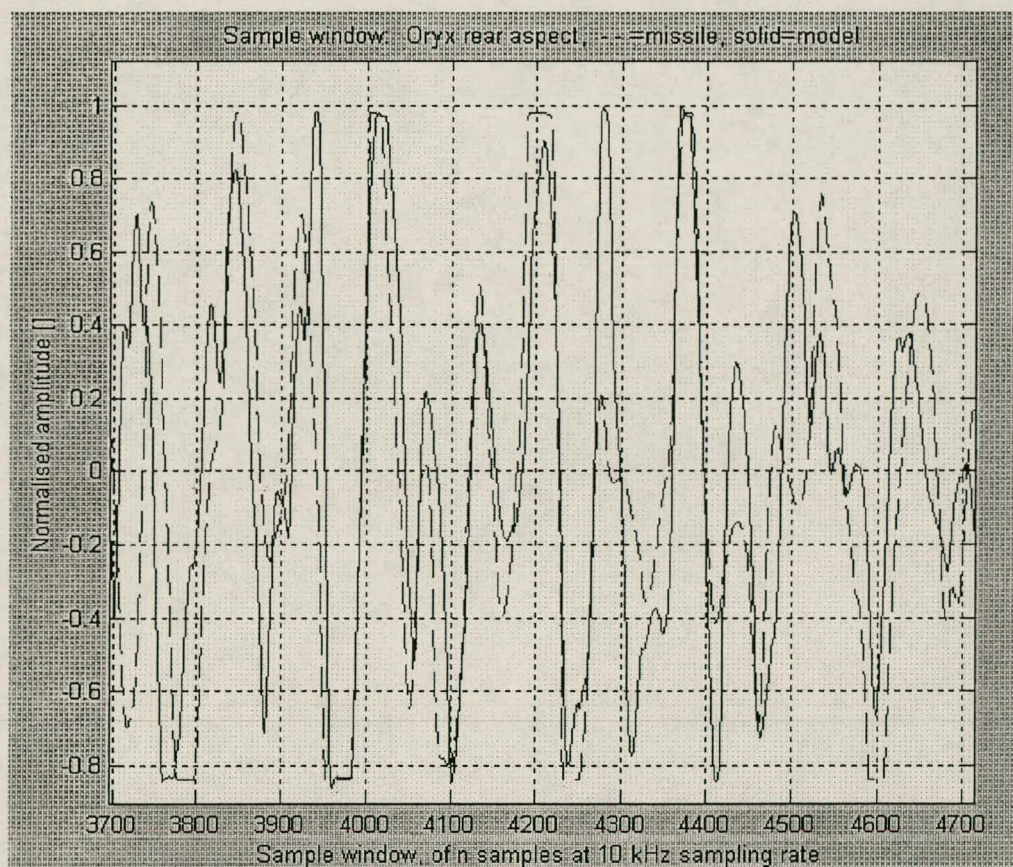


Figure 23 Missile response vs. pre-processing response for Oryx rear aspect view

5.2.3.1.5 Discrete model implementation

As a whole, the optimised pre-processing system by itself seems to be an improved model in comparison with any of the previous attempts. Implementing it in real time inside a computer

requires that the model be Z-transformed²³. Each of the two²⁴ ZOH-transfer-function combinations are Z-transformed using Matlab. The transfer functions are derived in state-space format from the block diagrams and optimised parameters of the Simuwin model. Matlab's "c2dm" function is used for the transformation, and the resulting z-plane state-space representation is then transformed into a transfer function using "ss2tf". These transfer functions are implemented directly inside Simuwin, as the program contains a module in which z-plane transfer functions may be entered as a proper numerator and denominator.

Unfortunately, the accuracy of the system falters after the z-transformation is done. This is due to the high order of the system, while the sampling rate of the input signal is only 2 to 5 times that required by the Nyquist principle. The combination of the LPF (Figure 19) and the precession amplifier/filter in Figure 20 only allow low frequency signals to pass. This implies that the output of this Z-transform combination is not affected as much as the differentiator structure's response after transformation, with the precession output being 15 to 50 times within the required Nyquist sampling rate. Comparison of the signals directly prior to the LPF structure within the discrete and continuous systems show major differences that may only result from inaccurate differentiator operation after transformation. As the input signal's sampling rate cannot be increased, it seems logical to revert back to the optimisation routine, this time in the Z-plane. Experiments with optimising all the system parameters quickly indicated counter-productive results, since three days of optimisation are required to realise that the filter forms after transformation are seriously distorted and ill-optimised. Keeping in mind that the inaccuracy problem arises from the differentiator, it was decided to re-optimize the differentiator structure only, without interfering with the Z-plane LPF/precession structure.

Figure 24 provides an illustration of the z-transformed pre-processing structure, with the two different z-transform structures indicated. All eight of the gain blocks within the Differentiator structure are optimised with a similar approach (Figure 18), using Simuwin's linear bisection capabilities. After optimisation, the pre-processing structure is implemented in exactly the same

²³ Various techniques exist, but usually require that integration step sizes etc. be performed at least 10 to 20 times faster than any relevant signal frequencies. The Z-tfm arena allows processing at the current available sampling rate of 10 kHz.

²⁴ i.e. The differentiator, and the LPF-precession-filter combination

form as shown in Figure 24, only with the newly optimised gain values imposed within the block diagram structure.

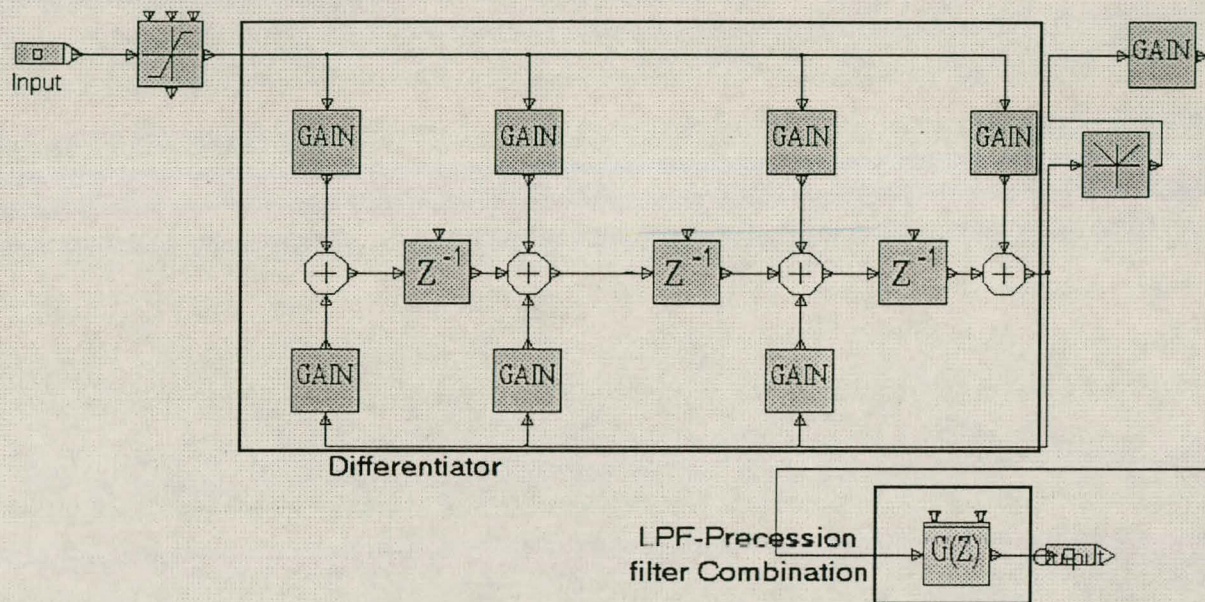


Figure 24 Z-transformed pre-processing structure

With the pre-processing structure now completed, it becomes necessary to process all the available IR detector signal samples into first iteration precession signals. These precession signals, together with the recorded missile precession responses, are then used to train the final neural network. Figure 25 illustrates a block diagram of the Simuwin structure used to process the IR detector data. It must be noted that the initial ZOH block at the system input will not be implemented when the system is eventually hard-coded and combined with the neural network. The final ZOH and anti-aliasing (AA) blocks will also not be used during HIL simulation.

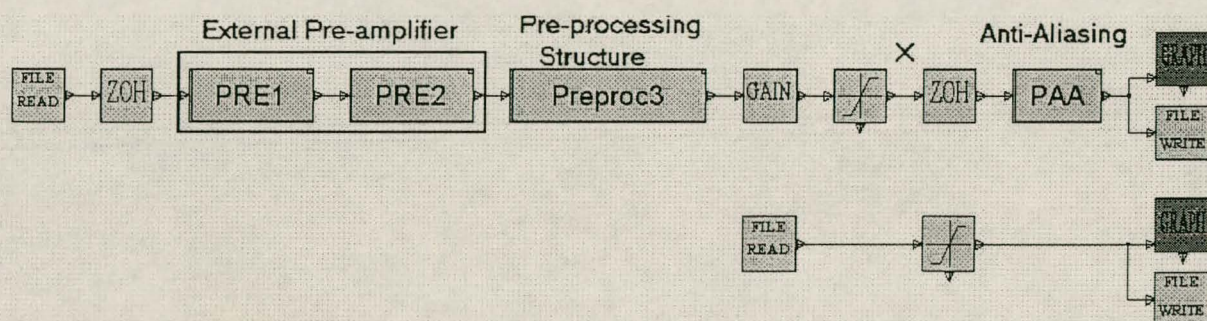


Figure 25 Final pre-processing structure prior to neural network training

The “X” in Figure 25 marks the closest equivalent of the true precession signal that a network can expect. Removing the AA filter and the ZOH blocks would see the neural network attempting to mimic their effects, since the recorded missile precession responses have indeed been influenced by such systems. The inclusion of their effects at this stage is part of an attempt to be able to “survive” without their influence once the HIL system is implemented.

It should also be noted that two file write blocks sample and save the pre-processing output as well as the recorded missile output data. This is done because the clock interrupt driven sampling system which recorded the data does not sample at exactly the same rate as the Simuwin system would when writing to file. The 8254-type onboard PC counter operates at 1.19318MHz, allowing us to set clock interrupt times at interval steps of 838ns. The closest attainable frequency to 10kHz when recording was therefore 10.0267kHz. Such a very small difference in sampling rate results in for example the 20 000th sample of the original precession recording and the pre-processing output not being representative of the same point in time, with relative phase shifts becoming evident. This time reference drift would continue getting worse towards the end of each 8.3 second input series, making much of the data useless for input-output comparison neural network training. Due to the extremely small difference in sampling rate, re-sampling the recorded precession signal in Simuwin at 10.0267kHz has very little distortive effect on the signal.

Figure 26 provides examples of the typical pre-processed outputs of 4 of the data series in comparison with their respective recorded missile precession signals. The simplest form of input, series a1, a 100Hz signal FM modulated onto a 1200Hz carrier, clearly indicates a fixed phase distortion. This phase distortion is clearly frequency-dependant, as indicated by the remaining sub-plots. It has been stated before that the phase relationships between the input and output of the system is crucial, as is the amplitude. The neural network structure is now left with the task of completion of the phase correction of the precession signals. Also of interest is the accuracy with which the model output represents the missile’s response to noisy “real” input signals such as series s1 and s2, that represent C130 rear aspect approaches.

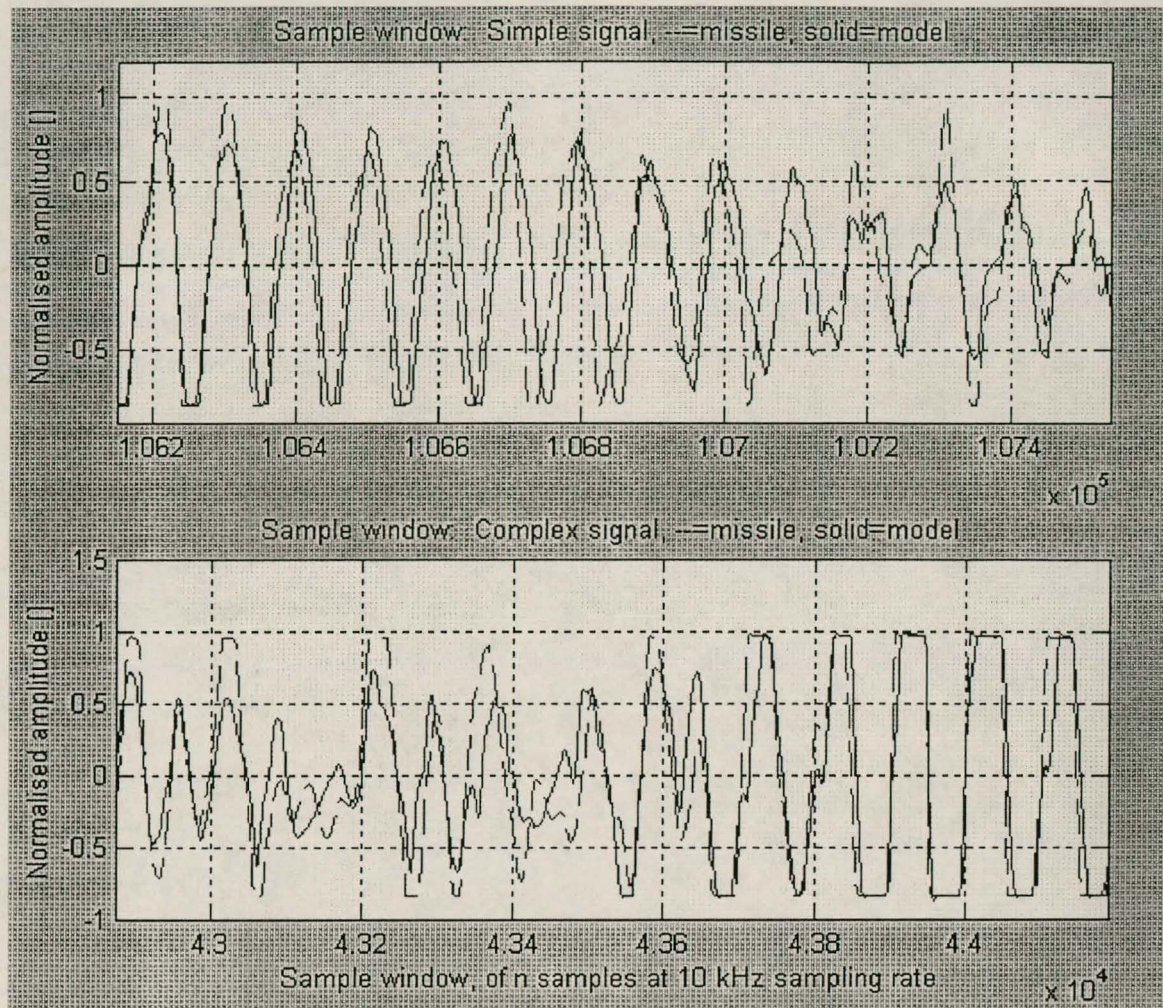


Figure 26 Typical final Z-plane pre-processing results

5.2.3.2 Final neural network training and theoretical analysis

5.2.3.2.1 Network implementation

With all the input data gathered in Chapter 4 pre-processed by an optimised pre-processing structure (5.2.3.1), modelled precession signals and a measured precession signals are now available. It the role of the neural network structure discussed in this section to convert this modelled precession response into its required form. This will create confidence that the missile model in its entirety is able to act as a valid representation of the missile tracking loop.

The output of the pre-processing system is limited in amplitude to the same extent as the recorded signal, with saturation occurring above the normalised amplitude of 1 and below that of -1. This is of importance to the network, as uniform network inputs with clear absolute maxima and minima allow the network to be trained to respond to signals within a fixed amplitude envelope. This can be completed without the need to cover the network's response

to high amplitude signals. Once more it is clear that the less flexibility is required of the network, the more freedom it has to focus on its flexibility in other, more important areas of operation.

All the pre-processed data is assembled in large matrices containing row vectors of pre-processing outputs 10 samples apart, with each time step's corresponding required output placed at the end of each of these vectors (5.2.2.2). Once again, the well proven multi-layer perceptron structure with sigmoid activation functions is used. SCG and RPROP training provide almost identical results. Figure 27 provides an illustration of the results obtained using RPROP on neural networks with different size architectures. These are but a few of the size architectures that were examined, providing insight into the final choice of network size. After 1000 epochs it can be seen that the 20X20X1 network out-performs the previously used 30X15X1 network in terms of MSE on training data. Of all the architectures examined, the 20X20X1 network provided the smallest MSE. Figure 27 also examines network sizes around the 20X20X1 mark, including 20X10X1 and 20X30X1 networks. It is noteworthy that networks with both more and less hidden neurones than the 20X20X1 structure have larger MSE's. Networks with both more or less input neurones also have larger MSE's. The 20X20X1 network therefore provides a local MSE minimum, within less optimal size surroundings. *It is crucial to note, however, that all of the networks within Figure 27 provide smaller MSE's than any results achieved thus far, making "less optimal" a relative term!*

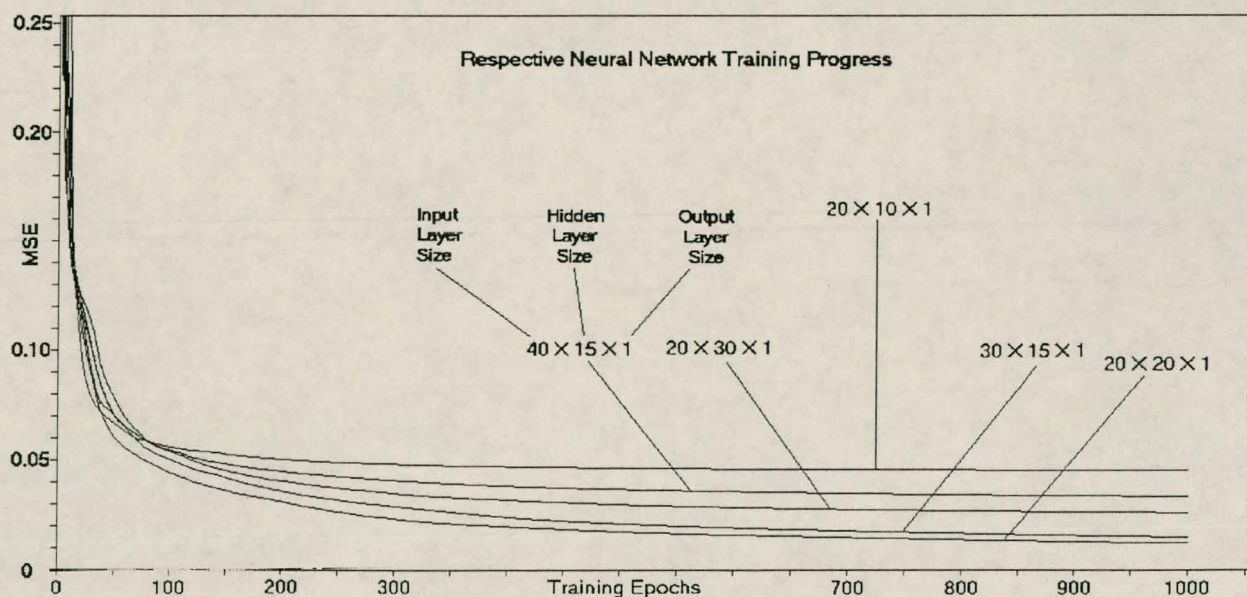


Figure 27 Respective neural network training progress

It should be kept in mind that Figure 27 provides a glimpse at only the first 1000 epochs of training. The 20X20X1 network delivers no discernible decrease in MSE after about 10 000 epochs. Taking into account that there are 24 000 pattern sets with which the network is trained during each epoch, 10 000 training epochs would seem to be too few. Pattern recognition heuristics indicate that the number of training epochs almost always outnumber the training pattern sets. In this case, however, many of the inputs to the network are repeated to a large extent, only shifted by 100 μ s after each pattern input. This effectively results in less input patterns than the stated 24 000. This necessarily small shift in input pattern is motivated in 5.2.2, providing increased time resolution accuracy to the trained network. With the small time shift kept in mind, it is understandable that traditional heuristics do not always apply to temporal neural processing.

A further interesting occurrence after 10 000 epochs is that the 30X15X1 network evens out to an MSE of 0.009 while the 20X20X1 network eventually provides an MSE of 0.008. The true difference in MSE here is minimal, implying that a decision made about the size architecture to be used may not be fully based on the MSE sizes of these two networks. The following points provide additional motivation as to why the 20X20X1 network was decided upon:

- The 20X20X1 neural network provides the smallest MSE when fitted to the training data.
- The 20X20X1 network requires less input pattern memory when implemented, especially in light of the inputs being representative of every 10th previous pre-processing output. This implies that ten times as many inputs need to be memorised and processed than is presented to the network.
- Most heuristics claim that smaller networks are able to “generalise” with more accuracy when presented with totally foreign input data²⁵.
- The 20X20X1 network is slightly smaller than the 30X15X1 network (i.e. less inter-neurone connections, which should provide an additional processing time buffer in our pursuit to eventually accomplish real-time processing HIL simulation).

²⁵ A statement to be viewed with some scepticism when dealing with temporal signals. Indications are that both the 20X20X1 and the 30X15X1 network fare almost exactly the same when “generalising”, with series s4 test MSE’s of 0.074 and 0.078 respectively.

Another concept that must be allowed for is **over-training** of the network. Over-training occurs when the MSE of test data/patterns reaches its minimum, while the MSE of training data is still decreasing. Further training results in the increase of test data MSE (i.e. decreasing the eventual network accuracy while only optimising the network's response to one single set of training data). It is crucial that a network responds as accurately as possible when presented with typical test data, not solely training data. Training data is but a tool to optimise the network's ability to process test data.

Results show however, that both the training data and the test data MSE settle at the same course in terms of the number of epochs trained. As the training response MSE settles to 0.008, the network parameters change very little. With the test data settling at the same time, the network test data MSE does not increase as the network is trained for more epochs. This type of behaviour is not unheard of when dealing with neural networks, usually being indicative of one or more of the following:

- The network training data is an accurate representation of the test data. This may be a problem when assembling all of the test and training data from the same data series, but in this case almost half of the test data is derived from series that are not directly related to training data.
- A large training pattern set is used, covering a large number of possible inputs in conjunction with a relatively small neural network. In a classification problem, this would indicate that most or all of the pattern classes which can be identified by the network are represented in the training data.
- The trained network structure is a good representation of the problem at hand. Once more, in terms of a classification problem, it may be explained thus: linearly separable pattern sets may be accurately processed by networks with linear activation functions. In the same way, the problem at hand seems to be well suited to the multi-layer perceptron structure with sigmoid activation functions.

5.2.3.2.2 Model analysis

With the MSE data now available it is known that the optimised pre-processing structure combined with the neural network delivers the smallest MSE of the structures analysed during the course of this project. But the following questions remain:

- Is the 0.008 MSE sufficiently small to brand the model as an “Accurate Model”?
- What does the output of the network look like and how does it compare to the desired outputs of the model? MSE’s are good indications of error energy, but they are no substitute for physically looking at the temporal data fit along the different series of inputs and deciding for oneself whether the output signal is “acceptable”.

None of the above-mentioned questions have clear-cut answers. However, a helpful way to quantify MSE is by calculating the RMS of the desired output signal and comparing it to the root of the MSE. In this way, the average absolute error (AAE) may be compared to the RMS signal amplitude and expressed as a percentage of its value. This is called the Relative Error Percentage (REP), or mathematically speaking:

$$REP = 100 \times \frac{\sum_{n=0}^N |y_{missile}(n) - y_{model}(n)|}{\sqrt{\frac{\sum_{n=0}^N (y_{missile}(n))^2}{N}}} \% \quad \text{Equation 5}$$

This is the first text to introduce the REP term as a means to quantify neural network output errors when representing temporal signals. A percentage of 0% indicates that the network is able to model the missile tracking loop without error, while 100% indicates that the network output error energy is identical to the amount of desired signal energy²⁶. The complete model constructed in this section has a REP of 22.14%, implying that the average network output error amplitude is 22.14% of the desired RMS output signal amplitude. This is about a third of the 64.2% REP obtained with the non-optimised pre-processing combined with a neural network (5.2.2.2).

Another well-known method of analysis would be to determine the error mean and variance. These are -0.00132 and 0.0303 respectively. With the desired RMS output having a value of 0.603, it can be seen that the network errs nearly exactly as much to the negative as to the positive, hence the small mean error. The variance of 0.0303, or standard deviation of 0.174,

²⁶ Please note that percentages higher than 100 % are possible, indicating VERY poor network accuracy.

serves much the same purpose as the REP indication, so long as the RMS of the signal is known (being 0.603).

Eventually, what the question comes down to is whether a REP of 22.14% and an error variance of 0.0303 is acceptable. This can be answered by looking at the typical temporal signal comparisons in Figure 28. Figure 28 may also be compared to Figure 26 to illustrate how well the neural network is able to transform the pre-processed signal into its required form. Very little phase error is visible in any of the series responses in Figure 28. The signal amplitude fit is also clearly quite accurate.

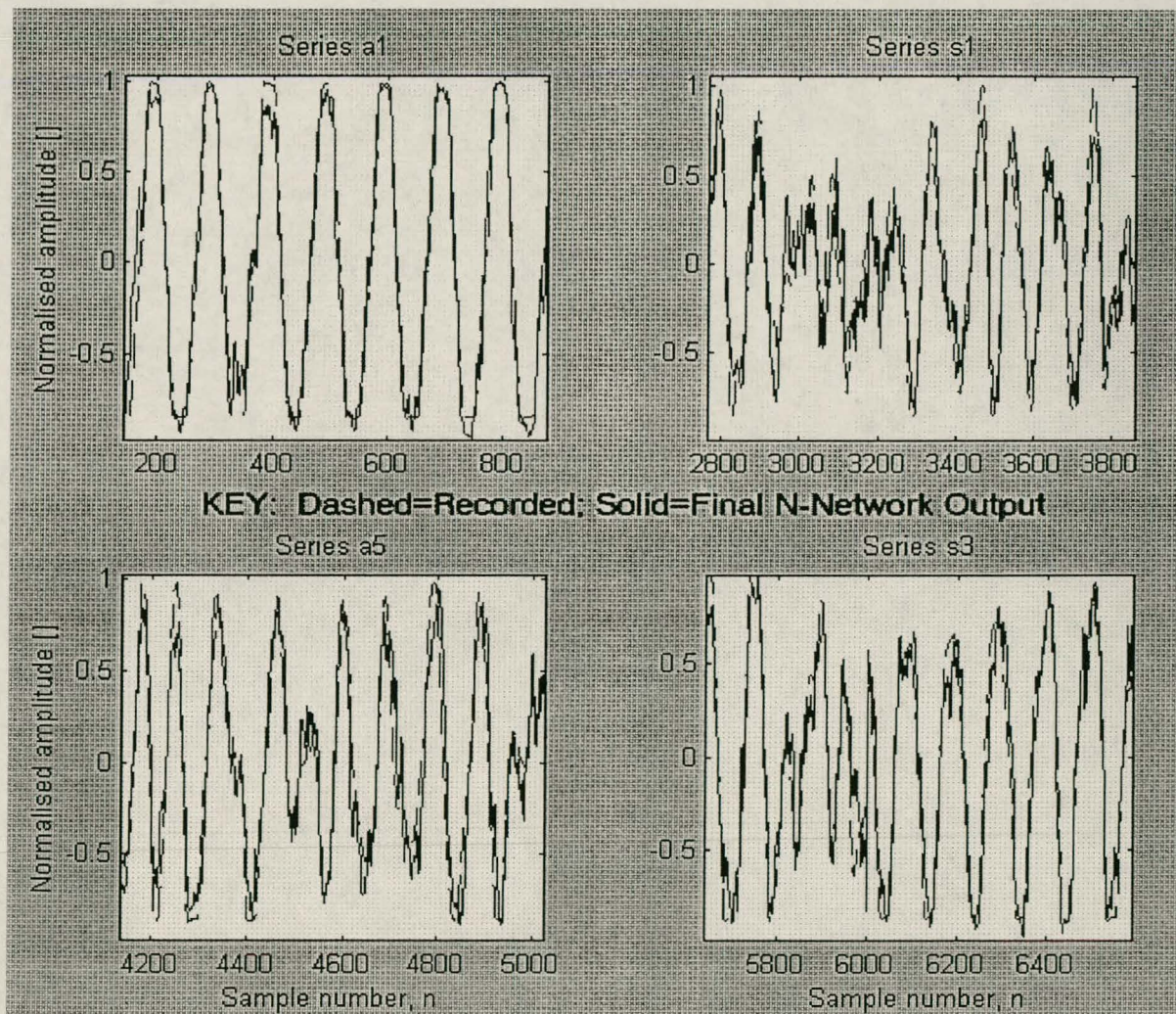


Figure 28 Optimised pre-processing & neural network combination training outputs

A high frequency AC noise signal with low amplitude is visible on the network output. This is caused by the network's output variance, as every network output sample error "randomly"

surrounds the required/recorded signal waveform. Each of these random errors are made at 10kHz (i.e. every time the network produces an output). The low output error mean of -0.00132 indicates that the network's output average lies close to the required signal, and the low variance assures that the high frequency noise on the output has a low amplitude.

Having now explored the temporal aspects of the model output as well as the cumulative error results, Figure 29 indicates what the typical frequency content similarities are between the model output and the corresponding missile response.

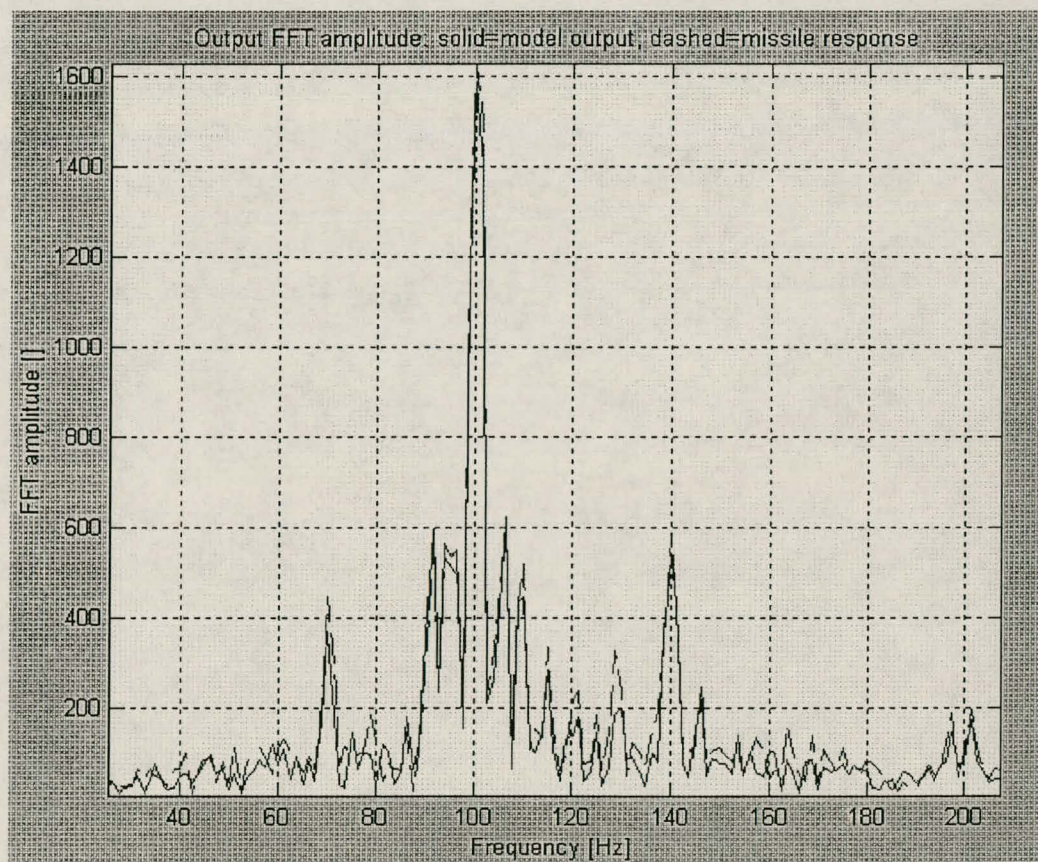


Figure 29 Comparative training data frequency response of model and missile

Only very subtle differences are visible between the missile and model outputs' frequency content. Figure 29 is a close-up over the typical precession signal frequency range. Although not indicated here, some added random and uncorrelated high frequency noise of very low amplitude is visible above 2kHz, which is explained by the discussion regarding Figure 28. The signal recorded to represent the precession signal was, in fact, not sampled at the precession

coils themselves, but on a slightly isolated signal test point known as the “ $\frac{4}{7}\sigma$ ” point. The precession coils are mainly inductive and resistive, therefore acting as a first order LPF. The gyroscope also acts as a natural mechanical LPF. Taking these additional LPF systems into account, both the missile and the model outputs will undergo filtering before steering the gyroscope. Any additional high frequency noise on the model’s output should have little effect when passed through the precession coils and effected on the gyroscope. This will work providing that its frequency is higher than the LPF cut-off. No attempt is made here to quantify the gyroscope’s filtering action, but the coil’s response may be characterised.²⁷

The precession coils’ inductance and resistance are measured using a LCR-meter²⁸, without removing the magnetised steel gyroscope assembly within the coils. The time constant, determined by $\tau = L/R$, indicates a LPF corner frequency of 1.8kHz. This is only an indication arising from the coil itself, and may vary according to the output resistance of the driver circuit and the orientation of the gyroscope within the coils, as changes in the gyro-coil flux-coupling influence the coil inductance. As stated above, very little noise is visible on the model’s output, and only at frequencies above 2 to 2.5kHz. The precession coils evidently have a LPF corner frequency at 1.8kHz, suggesting some rejection of the existing noise, even though it is only first order rejection, at -20dB per decade starting at 1.8kHz.

²⁷ See Appendix A

²⁸ The exact values of inductance and resistance may not be disclosed in this report.

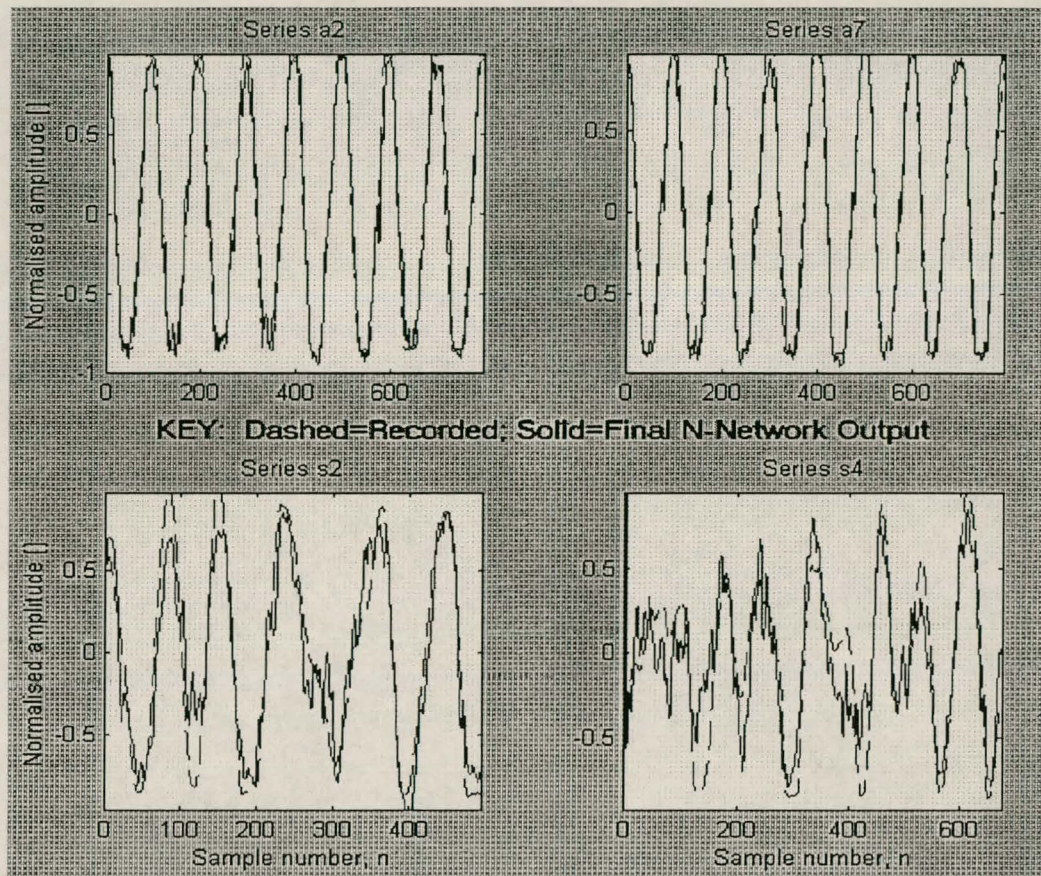


Figure 30 Typical optimised pre-processing and neural network combination test outputs

Figure 27 to Figure 29 indicate the model's performance when dealing with training data. The final part of this chapter deals with the model's performance when dealing with test data. The standard test data series discussed in Chapter 4 is used, consisting of data derived from the same series used during training, only over different time intervals. Completely foreign data such as series s4 (a panned Oryx helicopter rear aspect view) is also used. Figure 30 indicates the model's response to such test data.

Series a2 and a7 are relatively simple missile responses to artificially created FM inputs (i.e. not created from flight scenes). Series s2 and s4 provide an account of the model's abilities to represent the missile's response to proportional C130 approach with a horizontal horizon background and an Oryx helicopter respectively. It is clear from Figure 30 that the outputs to series a2 and a7 are represented with acceptable accuracy. Both series s2 and s4 were not used during neural network or model training, therefore providing a good indication of the model's

response to unfamiliar inputs. Some model error is visible for both of these series, but the required signal trend seems to be followed.

These two series alone contribute an MSE of 0.05, compared to the 0.02 MSE of the entire training set, and the 0.008 MSE of the training data. The 0.05 MSE of the unfamiliar input series should represent the worst case scenario of the model, especially in light of the HIL model implementation taking place on a single point source target simulator. The test signal error variance only increases to 0.036, compared to the 0.0303 variance of the training signal error. The error variance of the series s2 and s4 combination increases to 0.0544, resulting in a standard deviation of 0.233.

While it is always desirable to have the smallest possible MSE, only the HIL test itself can now indicate whether the results attained thus far provide a sufficiently accurate model. Various questions also arise regarding the difference in precession signal that would result when two seemingly identical missiles are confronted with a series such as s4. Missile component tolerance, and small differences in construction might easily cause MSE's around 0.05 between two such missile outputs, with both missiles operating and tracking within specification. The answers to these questions are as yet unknown.

A further remark regards the nett difference in gyroscope movement when slightly different precession signals are effected upon it. It was indicated previously that the gyroscope and the precession coils each provide an order of LPF. If we assume that the gyroscope's LPF corner frequency is approximately 800Hz²⁹ and that of the precession coils approximately 1.8kHz, the precession signals in Figure 30 may be filtered. The results of this action are shown in Figure 31.

²⁹ See Appendix A for details of this frequency choice

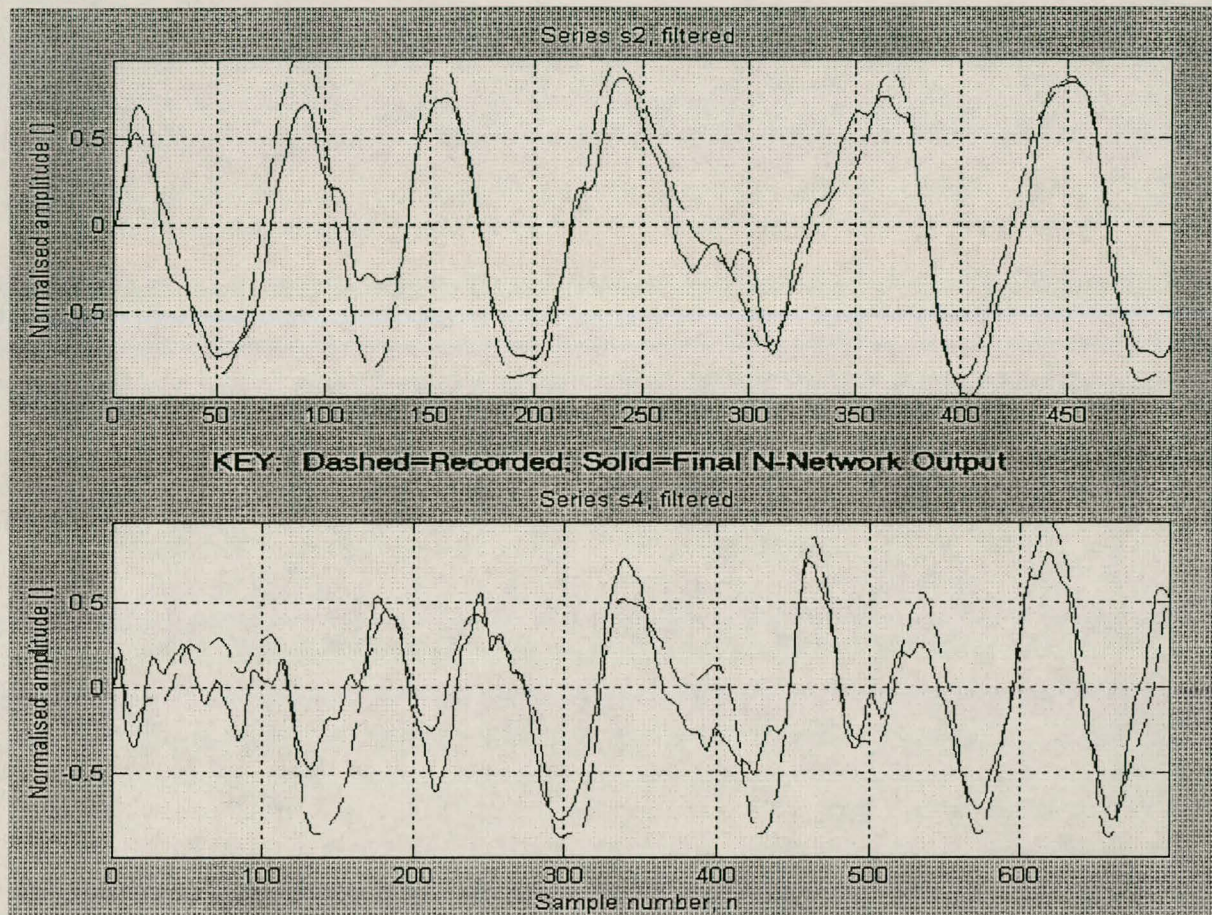


Figure 31 Series s2 and s4 relative responses after gyro and precession LPF action

The filtered series s2 and s4 responses in Figure 31 should provide a clearer picture of the nett difference between the model's precession output and the recorded missile response. The MSE of these two combined series decreases slightly to 0.041 after filtering. It is the opinion of the author that even if Figure 31 provides some of the worst possible results attainable during HIL simulation, the model should be sufficiently accurate to obtain desirable results when its target tracking capabilities are compared to those of the missile's tracking loop. It remains to be seen whether this will be the case. Chapter 6 deals with the HIL simulation set-ups and results.

CHAPTER 6

MODEL IMPLEMENTATION AND HIL SIMULATION

6.1 Introduction

Having now constructed and theoretically evaluated a model of the missile tracking loop, verification of its efficiency rests on the implementation of a HIL simulation. Illustration of the modelling concept remains solely theoretical, but its physical implementation in the place of the missile control electronics will provide the opportunity for necessary practical evaluation. However, prior to this implementation, various factors require further attention.

Chapter 4 dealt with the only hardware-related problem thus far, namely the signal injection and recording system approach. An artificial signal was injected into the missile via signal injection hardware, and the resulting missile precession signal was recorded. The control loop, in which the orientation of the gyroscope with respect to the target LOS completes the loop as it determines the IR detector signal, was never closed. A pre-processed and recorded fixed detector signal input was injected, and no amount of electronics control response could influence this. This was undertaken to ensure the capture of the essence of the control electronics, with target lock and gyro dynamics playing no necessary role.

During HIL simulation, this scenario changes drastically. The aim of the HIL simulation is the verification of the target tracking abilities of the model. These must approximate those of the original missile, necessitating feedback from the gyro-target LOS relative angle onto the detector signal.

It was thus decided to remove the front end of the missile (containing the IR detector, gyroscope, and missile optics) from its seeker assembly, and to make use of an optical target scene simulator to simulate a true InSb detector via the missile optics. Any control effected upon the gyro would then influence the IR detector signal directly. Several serious hardware considerations do however arise from this.

This chapter assists in the validation of the model. The model is aimed for use in total software only, and the HIL implementation is therefore not restricted by several of those

boundaries associated with rapid modelling which were imposed during model construction. However, since this type of model is a first, theoretical validation is insufficient. The practical implementation of the model through the HIL simulation is an attempt to surpass this shortcoming, and provide effective proof of the validity of this model for use under these circumstances.

Before proceeding, it is useful to note that a block diagram of the HIL simulation system including all the hardware and software is available in Figure 33 of section 6.2.4.

6.2 Model Implementation Hardware

6.2.1 Missile front end

In essence the missile front end consists of a collection of “dead” hardware which requires activation before it can be of any use.

- The gyro has to be spun up to its operational speed of 100 rotations/s. This is achieved by re-assembling the gyro spin regulation electronics once removed from the missile. The re-connection of these electronics is completed in accordance with schematics obtained from disassembly of the missile by Aerotek personnel. Reassembly of the electronics is relatively straightforward, requiring only two inputs from the front end, and DC power. The first of these inputs is: a gyro rotation reference coil output, supplying the gyro spin rate relative to the missile body. The second is a cage coil output, supplying gyro look angle information and its spin rate relative to the horizon. An amplifier and spin drive provide the increased power necessary to control the gyro spin rate via two pairs of spin coils surrounding the gyro.
- The InSb detector requires biasing at one end with a DC voltage before it is able to serve as a detector. The detector acts as a variable resistance depending on the amount of IR radiation reaching it. It also provides a varying voltage over a fixed resistance connected between its remaining end and the signal ground.
- The gyro requires caging towards the missile's centre axis, in order to provide a gyro positional reference from which the simulations may be started. As it was developed for use in Appendix A, the caging system is not included in Figure 33.
- The InSb detector is cooled to $-200\text{ }^{\circ}\text{C}$ in order to suppress its background noise levels. Under normal circumstances, the detector would be cooled by high pressure nitrogen gas venting onto an aluminium header directly behind the detector. The gas supply system is

reconnected to the front end and supplied by a 42 MPa nitrogen tank. The detector must be cooled before use! *Upon cooling, the detector impedance rises from about 2Ω to nearly $3k\Omega$. This is taken into account by the preamp.*

6.2.2 Missile electronics

As stated in the introduction to this chapter, certain leeway regarding rapid modelling philosophy is permitted during this test phase of the target tracking loop model. This is due to the fact that this HIL simulation is only completed for additional model validation purposes. In 6.2.1 rapid modelling boundaries had begun to be crossed through the disassembly of the missile front end, and reassembly of its gyro spin regulation controls. The spin regulation control system is made up of three separate modules within the missile, and is not the only missile electronics necessary for the HIL simulation:

- It was first stated in 5.2.3.1.2 that the missile's preamplifier would be necessary during HIL simulation, as the biased InSb detector delivers very low power output signals. It would be possible to construct an amplifier that could perform the same function, but the result would be a less optimal model, since the differences between this new amplifier and the true preamplifier would have to be modelled. It would also prove an unnecessary difficulty, as the preamplifier is contained within an easily reconnectable module. Since discussion in 5.2.3.1.2, the preamplifier has been included in the modelling architecture. This implies that it would also be necessary to include it during HIL simulation in order to account for its filtering and amplification effects etc.
- Steering (precession) the gyro requires a power amplifier. The set-up within the missile includes a push-pull twin power transistor driving circuit for this purpose. A transformer splits the incoming precession signal into two signals -180° of phase apart. This implies that only one transistor switches on at a time for either the top-half or the bottom-half of the original precession signal's cycle. For example, a sinusoid signal with 20 V amplitude and 0 V offset would switch on the top transistor in Figure 32 for the positive half of its period cycle, and the bottom transistor would switch on during the negative input voltage cycle. This type of design allows the full -42 V power supply to be used for each half of the incoming signal's amplitude, effectively doubling the available output power band to the precession coils. It is important to recall that the two sets of precession coils wrapped around the gyro are wrapped in different directions to be of use when driven in a push-pull

manner. The function of the transformer in Figure 32 has been replaced by an inverting and non-inverting operational amplifier pair with unity gain, as the precession amplifier always requires two 180° phase split signals. Note that these Op-Amps form the “Phase Splitter” in Figure 33, capable of delivering 20 mA to the driver circuit, which is the maximum required precession driver input current.

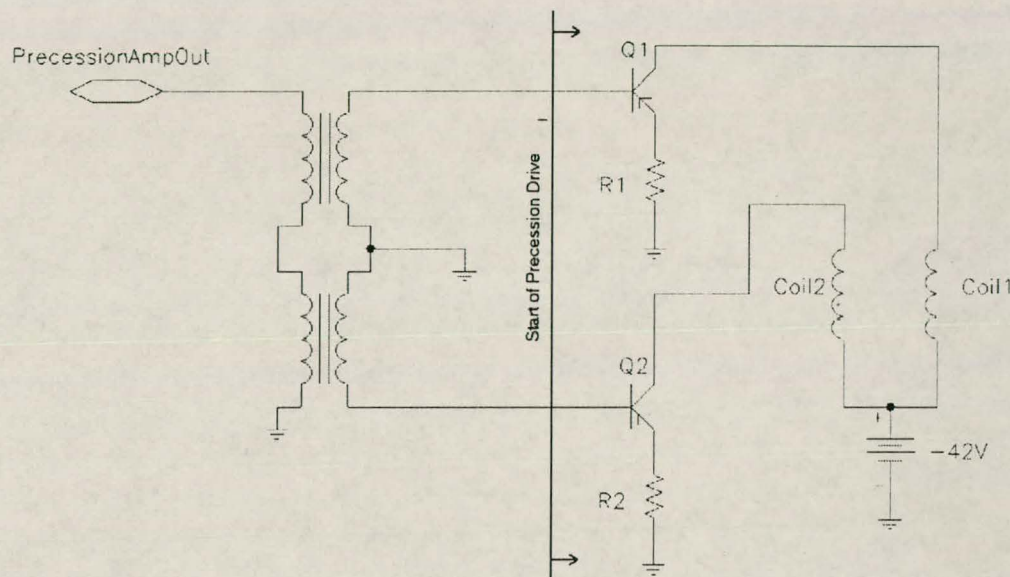


Figure 32 Basic precession drive schematic

6.2.3 A/D and D/A hardware

During HIL simulation, the “ADAS”-card introduced in section 4.2 is used. Here however, the card is used to sample the output of the preamplifier through an A/D converter channel, and to write out precession information to the phase splitter via a D/A channel. An anti-aliasing filter (AAF) is present on the card’s A/D converter with a LPF corner frequency of 1048Hz. The output of the preamp contains relevant frequency information exceeding this corner frequency limit. The AAF is adjustable, but moving the corner frequency to about 2kHz does not allow enough rejection of the frequencies above the Nyquist folding frequency (i.e. 5kHz because of the 10kHz sampling rate). Simuwin simulations with the AAF included also indicate that the phase distortion delivered onto the signal by the AAF is up to half a decade and more below its corner frequency, severely distorting the output pre-processing algorithm. These factors lead to the removal of the AAF from the A/D converter for HIL simulation purposes. But how is aliasing then handled? The pre-amplifier transfer function used in the model optimisation structure (Figure 18) indicates a BPF with a centre frequency of approximately 1200Hz and a -42dB signal attenuation at 5kHz. Unfortunately, the output structure of the preamp saturates

and limits its output if the input signal amplitude is slightly too large. This forms part of the gain control mechanism of the CSFM based tracking system. Saturation of this signal adds high frequency harmonics to the pre-amplifier output, causing aliasing when the signal is sampled at 10kHz. *Adjustment of the IR sources' radiation levels is needed in order to ensure that the preamplifier is not driven into saturation. This is done with ease through the regulation of the current supplied to the wire-wound resistors (acting as sources), and the checking for preamp saturation prior to the trial of HIL simulations.*

The final A/D and D/A converter considerations deal with impedance levels. The preamplifier typically drives a $6.2\text{k}\Omega$ load in the form of missile demodulation circuitry. This load is simulated by a $6.2\text{k}\Omega$, 1W resistor with the high impedance ($>90\text{k}\Omega$) A/D sampler connected directly onto the load resistor.

The D/A converter channel drives the phase splitter, which is buffered via an inverting Op-Amp configuration with a $10\text{k}\Omega$ input impedance. This implies that only 1mA of current is required of the card when outputting its maximum voltage of 10V. The card is capable of supplying current in excess of 20 mA through use of the Op-Amps connected to its output circuitry.

6.2.4 Target simulator set-up

The target simulator consists of a wire-wound resistor viewed through a pinhole with the radiation from the pinhole being directed towards the missile front end by means of a set of prisms and a collimator to provide a good approximation of a point source far field target. The resistor is mounted on a worm-screw driven by a stepper motor, which allows for moving target simulations. The pinhole at the source may be opened or closed with a solenoid controlled shutter. The entire target simulator assembly is mounted on a flat single axis rate table bed, with all the electronics; including the stepper motor, shutter and rate table controls being controlled by a personal computer.

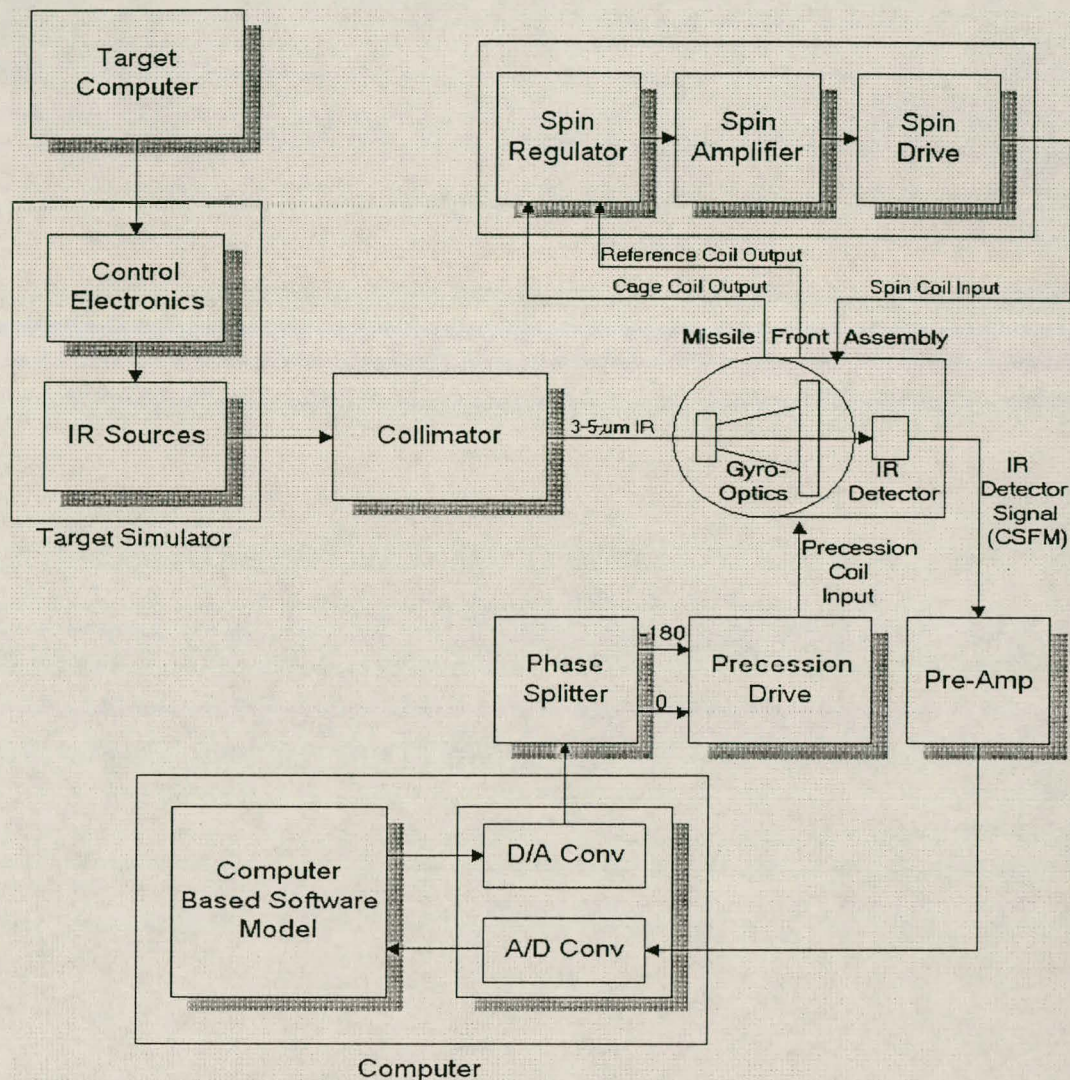


Figure 33 HIL Simulation set-up block diagram

6.3 Software

A summary of the programming architectures of the different modelling structures employed in the final missile target tracking loop model is given below:

- The Stuttgart Neural Network Simulator (SNNS) used to provide the neural network processing delivers its final trained network in the form of a “.net”-file, which consists of a text description of the network architecture, weights, biases, etc. A “SNNS2C”-executable is provided with the program which permits the “.net”-file to be ported to Linux Gnu-C or “GCC”. “GCC” is the standard Linux 32-bit C-programming language.³⁰ This C-code may

³⁰ SNNS runs under Linux.

be linked to a user program as a procedure call, requiring a pointer structure to the current network input and returns the network output response in a variable after being called.

- The pre-processing structures preceding the neural network were constructed in Simuwin and consist mostly of discrete Z-transformed transfer functions.
- The ADAS-card interface structures are available as Pascal routines that may be compiled under 16-bit MSDos.

With the SNNS neural network output file being available only in C and taking a quick look at its complex program listing in Appendix B, it would clearly be advantageous to write the remaining HIL software in C. Unfortunately, initial attempts at compiling the SNNS code in Borland C and Topspeed C under MS-Dos failed due to multiple structure declaration shortcut within the code that is apparently only supported under GCC in Linux. At first, attempts were made to rewrite part of the code in the SNNS20d.c file in order to bypass the necessity for the multiple declaration, but this proved ineffective and time consuming. The stumbling block was eventually overcome by acquiring GCC for the MS-Dos environment, as part of a package known as DJGPP. Being a command line C interpreter, the DJGPP package's GCC functions much like the Linux version. With the switch to DJGPP, a switch was subsequently necessarily made to 32-bit Dos, as this is the only mode supported by the package. This shift from 16-bit operation added another complication: *The protected mode environment*.

With the protected mode environment activated on the CPU, the standard timer interrupt assembly code used to time the A/D sampling and D/A writing becomes obsolete. This necessitated the writing of Timer.s (Appendix B), a GCC assembly language-based protected mode timer interrupt initialisation program and handler. The program makes use of operating system environment calls in order to switch between memory base addresses and segment offsets. This becomes a necessity when setting up the interrupt vectoring and chaining in protected mode.

The assembly code used to interface with the ADAS-card's A/D channels also had to be replaced by C code, as this assembly code would operate correctly only in the 16-bit Dos mode. The eventual HIL simulation program is listed in Appendix B. The following notes will be of use when interpreting the code:

- The program awaits a flag to be set by the interrupt handler; then runs through one A/D & D/A and processing routine; resets the flag and again waits for the flag to be set before the procedure is repeated. The program exits after 20s of simulation.
- Note the indicated assembly code which was removed from the “AtoD” procedure and replaced by its equivalent C code.
- A/D conversion occurs almost directly after the interrupt occurs, and is followed by D/A conversion of the output determined during the previous interrupt session. These operations are completed as close as possible to each other, in order to keep the A/D and D/A operations synchronised. Writing the output of the previous interrupt session to the D/A port during the new session approximates the final ZOH block in Figure 18. As processing time is not fixed and D/A and A/D conversion would “jitter” or lose synchronisation, this is a predictable and controllable alternative to D/A converting each new output at the end of each interrupt session (after processing).
- Although A/D & D/A conversion, pre-processing and neural network operation occur at 10kHz, the neural network receives the current pre-processing output together with the previous nineteen spaced 1ms apart. A rotating buffer array called “PROCHISTORY” holds the past 191 pre-processing outputs 100μs apart. A short FOR-loop executed during each interrupt cycle extracts every tenth sample from this buffer and places it in “netinput”, aptly named for being the 20-sample vector input to the neural network.
- The pre-processing structures developed in Simuwin and illustrated in Figure 19 and Figure 20 have been hard-coded in C exactly as the figures suggest. The state space implementation and accompanying sample history shifts when implementing the filters is noteworthy.
- “SNNS20d” is the function called by the program to carry out the neural network processing. The main HIL program operation comprises approximately 25% of the PC’s processing time, with the neural network accounting for the remaining 75%.

6.4 Target tracking results

This section considers the results obtained from three different types of HIL simulations. In each case, the simulation model was used to respond to a different target tracking scenario. The target control computer measured and recorded the gyro LOS of the missile front end. In order to make this measurement, the target control computer connects to the cage and reference coils together with the gyro spin regulation electronics. Upon completion of all three these target tracking scenarios, the HIL set-up was removed from the rate table and replaced by a fully functional seeker head. These scenarios were then repeated as inputs and the true response of the missile was recorded by the target tracking computer. In this way, comparison between the model's behaviour and the original missile's behaviour can be made.

The relatively basic nature of the target tracking experiments completed throughout this section must be kept in mind. It cost Aerotek CSIR valuable time and money, around 100 man-hours, the use of their rate table and target tracking equipment and high pressure scrubbed nitrogen in order to complete the following experiments.

6.4.1 Stationary target

This section includes much detail concerning the HIL implementation and missile-target response measurements. Further experimental procedures for step and rate tracking will be described only briefly, as they are variations of the experimental procedures in this section.

6.4.1.1 Experimental importance

This experiment verifies that the missile is able to remain locked on a stationary target directly along the current gyro LOS. This is an initial experiment aimed at recording the missile's behaviour under "zero-stress" target tracking conditions, which illustrates the model's natural mode. With the IR signal operational, the control loop is fully closed, making this the first closed loop response.

6.4.1.2 Experimental procedure

The following procedure is followed in order to record the model's reaction to a stationary target with a LOS lying directly along the missile's centre axis:

- Wire-wound resistor IR source is preheated and set at far-field along the missile's centre axis, representing the target.

- The gyro spin and caging electronics are activated, bringing the gyro to a 0° LOS rotating at 100Hz..
- The InSb detector is cooled to operational temperatures by a constant nitrogen gas flow.
- After approximately 5 seconds the detector has cooled sufficiently for the preamplifier to exhibit a clear FM output. The front end is now aligned accurately with the target until the preamp output transforms into a clear 1200Hz signal with minimal FM content.
- The IR source supply current is set so that the preamplifier output amplitude is only slightly outside its saturation limits. This step is time-consuming, as the sources retain their heat for quite some time after their supply current has been altered.
- The gyro and target LOS logging system is activated, being is a Hewlett Packard storage oscilloscope connected to the target control computer.
- The gyro is uncaged and the computer model is simultaneously activated, passing control of the gyro to the main HIL simulation program.
- After 20s the model deactivates itself and the gyro is manually caged. LOS logging lasts approximately 6s at a 5 kHz sampling rate before the oscilloscope runs out of memory.

The procedure is followed when determining the missile's stationary target response :

- Heat the infrared sources to the same temperature as with the HIL simulation.
- Cool down the InSb detector.
- Activate the missile power. This will cause automatic gyro caging and spin-up.
- Align the missile assembly to the target source until only 1200Hz carrier is visible on preamplifier output.
- Enable the missile "soft-cage". Soft-cage is a built-in missile mode which allows the gyro to return to the missile centre axis LOS when no IR target is apparent.
- Record the missile's LOS response.

6.4.1.3 Results

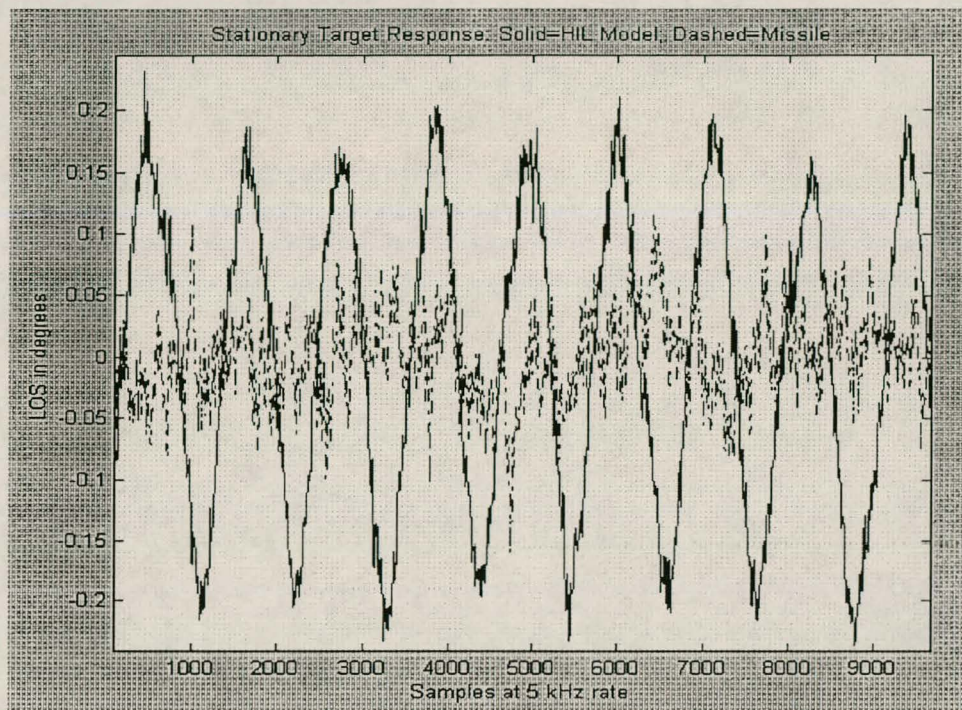


Figure 34 Stationary target HIL simulation vs. Missile relative LOS response

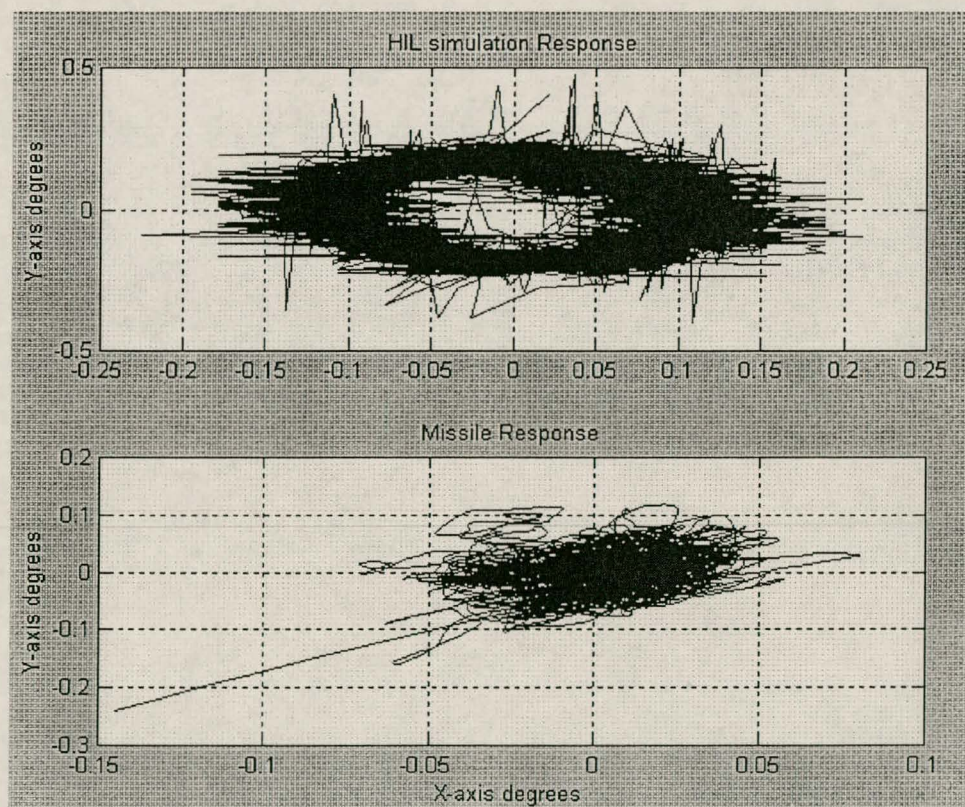


Figure 35 HIL simulation vs. missile, stationary target tracking pattern

Figure 34 provides an indication of the gyro LOS for the simulation and the missile over time. Although a clear oscillation is visible on the simulation output, the missile LOS exhibits much less variation in amplitude and no oscillation. The target control computer clearly indicates that both the missile and the simulation track the target during the experiment. This is also apparent from Figure 35, which provides an indication of the X-Y look angle covered by the missile and the HIL simulation during the course of each experiment³¹. Both of these measurements contain some noise, but the principle indication shows that both systems do remain locked on the target. The low frequency gyro oscillation causes the HIL simulation gyro to circle the target, without being able to settle.

No previous indication arose that such an oscillation would occur, but since this is the first time that the control loop has actually been completed, it comes as no surprise that such an oscillation is possible. The situation will be remedied. The gyro nutational oscillation frequency appears to have settled at approximately 4Hz, with an amplitude of approximately 0.3 degrees.

6.4.2 Step response

With the oscillation visible on the stationary target response, indications are that the gyro will settle into oscillation before and after completion of the step response. It should also have some influence on the step transient response.

6.4.2.1 Experimental importance

A target step response should indicate whether the missile and the HIL model are able to exhibit a similar transient response.

6.4.2.2 Experimental procedure

During this experiment, the missile and HIL model, both still aligned towards the centre of the target collimator, are initially caged towards their centre axes. The target is then moved 0.6° from the initial centre of the gyro LOS until the preamplifier output signal indicates that the target point source intercepts the first of the 6 centre spokes on the reticle illustrated in Figure 1. This is the point at which the target is scanned over the reticle approximately 50% of the time. It is scanned over the outside of the reticle FOV for the remaining 50%. The missile will

³¹ Consider the target as a location (0;0) on a grid, and the mass of lines in the graph representing the area around the target over which the gyro is swept.

regain central LOS lock on the target when the gyro is uncaged (i.e. a step response results). Most real-life target tracking remains within this 50/50 reticle scan region.

6.4.2.3 Results

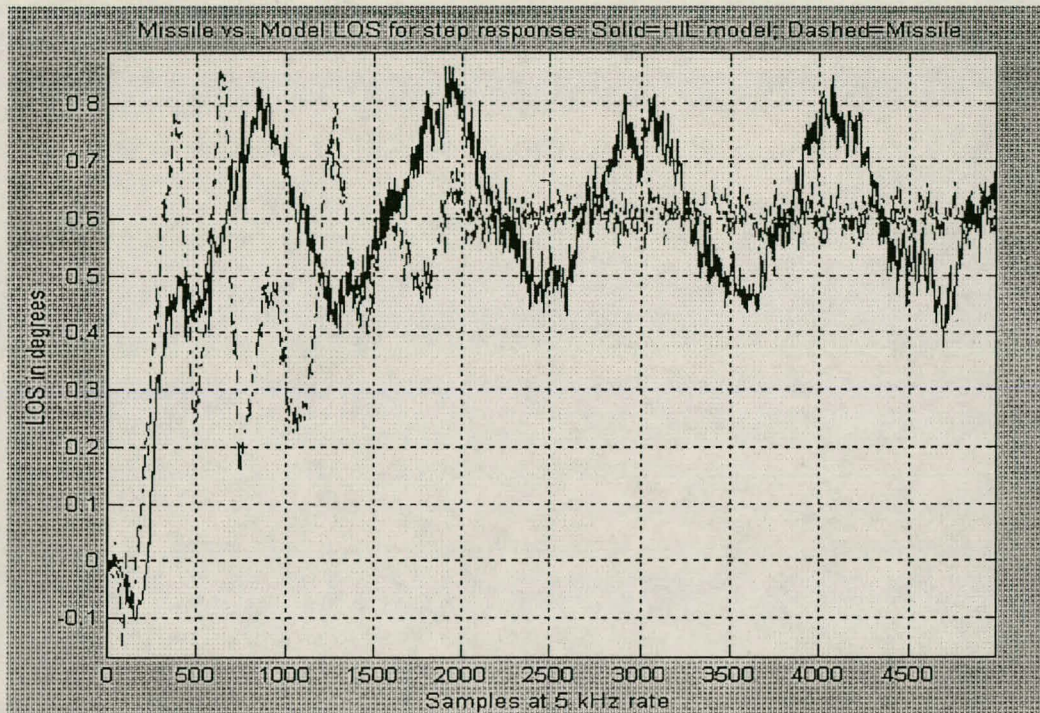


Figure 36 HIL simulation vs. Missile relative LOS target step response

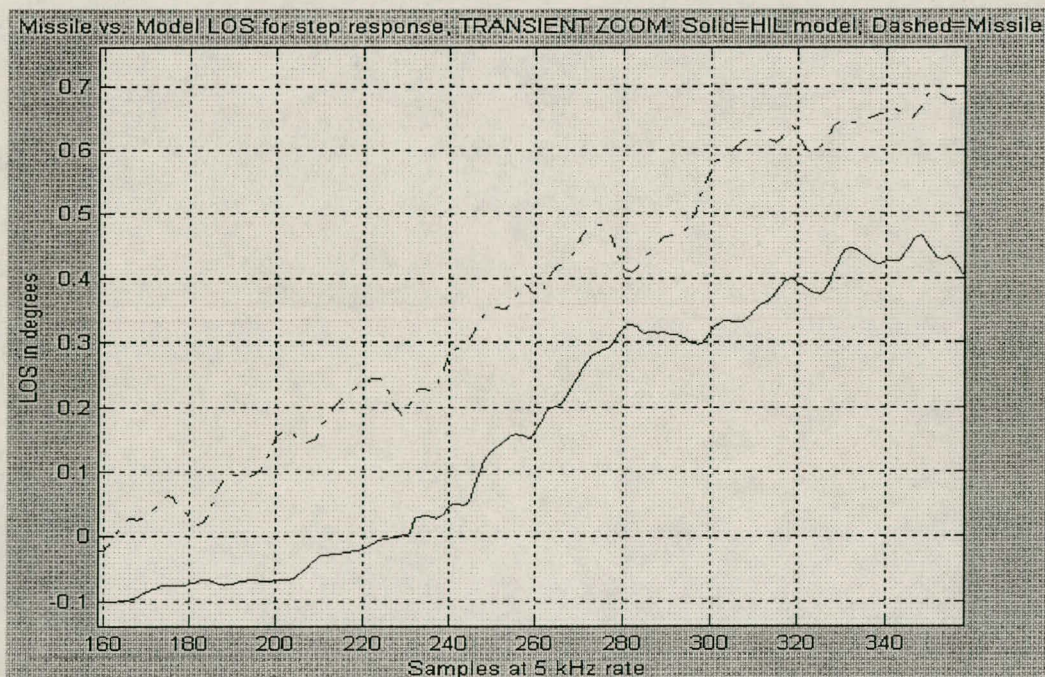


Figure 37 HIL simulation vs. Missile relative LOS target step response transient ZOOM

Figure 36 indicates that the model does indeed lock onto the target during the step response. The step response seems similar to the missile's response, barring the 12 Hz transient component visible on the missile's response. This is partially visible on the HIL model's response, but overpowered by the 4Hz component which re-emerges when the gyro approaches the direct target vicinity. Both systems initially track away from the target for a short interval before moving towards the target, which is characteristic of a non-minimum phase system. The oscillation frequency and amplitude is almost identical to the example in Figure 34, a veritable superposition of the stationary target response onto the step response. Finally attention should be drawn to the apparent slightly under-damped nature of the missile's transient. At this stage, it is unclear whether the model's behaviour would be equivalent, as the oscillation obscures its response and may quite easily influence apparent damping.

Figure 37 contains a close-up of the gradient, followed by the systems during their step transients. Both systems appear to indicate a tracking rate of approximately 20°/s during this transient. The 20°/s maximum missile tracking rate is well known, and it is encouraging to see that the model exhibits the same maximum tracking rate!

Note that a slight time shift is visible between the graphs in Figure 37, as the missile gyro polarisation vector is oriented differently at the outset of each step response. The gyro can only be precessed along a single axis, therefore a lag results when waiting for the gyro to rotate to a position where it may be precessed towards the target.

6.4.3 Rate tracking response

6.4.3.1 Experimental importance

Section 6.4.2.3 illustrated both the model and the missile's ability to track at approximately 20°/s during a step response. This section attempts to illustrate the model's ability to track a target moving at a fixed LOS rate with respect to the missile.

6.4.3.2 Experimental procedure

The missile and HIL simulation are allowed to lock onto a stationary target. After lock has been obtained, the target is moved at a fixed LOS rate by controlling the target stepper motor speed with the target control computer. The target is accelerated at its maximum ability to the

given rate. No uniform target acceleration is effected by means of the steppers, as this is not within the computer's current programmed abilities.

6.4.3.3 Results

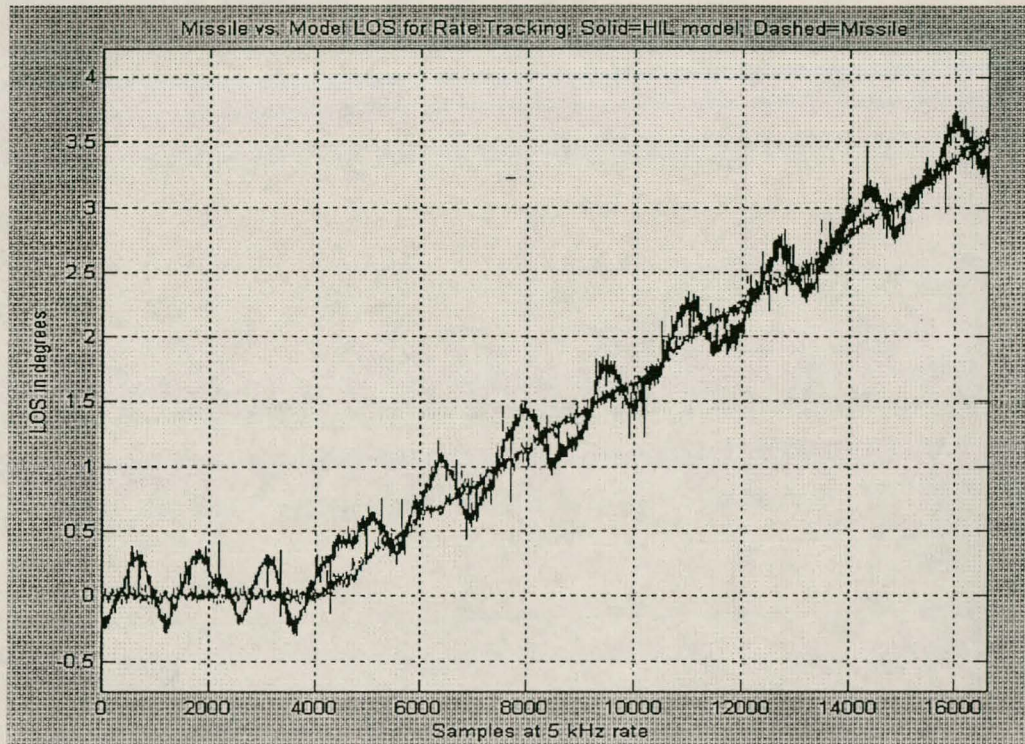


Figure 38 HIL simulation vs. Missile relative LOS target rate tracking response

Figure 38 indicates the model and the missile's response to the highest target LOS rate that could be consistently tracked by the model (i.e. $1.5^{\circ}/s$). Some success was attained around $3^{\circ}/s$ and even $5^{\circ}/s$, but the model frequently lost track of the target at these rates. Yet the missile is able to track at approximately $20^{\circ}/s$ during a step response. The answer to this discrepancy seems to lie in the 4Hz oscillation which is also evident in the model's rate tracking response: the largest precession signal attained from the IR detector input occurs when the target is swept across the narrowest part of some of the 12 spokes towards the centre of the reticle, while at the same time being scanned outside the reticle pattern 180° further along the line. Moving the target only 0.05 degrees further away from the centre of the gyro would result in the target being swept across some of the six central spokes, seriously reducing the precession signal amplitude. Because of the reduced precession signal amplitude, the missile cannot

support a $20^\circ/\text{s}$ tracking rate when this secondary area of the reticle has been reached³². With the target accelerating to its fixed LOS rate many times faster than the missile's gyro acceleration abilities, the target vs. gyro LOS error initially increases and then the gyro settles to a smaller fixed tracking error, depending on the target LOS rate. The missile itself loses target tracking if the tracking error is such that the target moves over into the secondary reticle area before the gyro has reached a sufficient tracking rate. This occurs with a target rate test of around $18^\circ/\text{s}$. With the evident oscillation on the model's output having an amplitude of approximately 0.3° the HIL model's target moves into the secondary section much sooner. This seriously inhibits the model's performance during the rate tracking test!

The crux of the matter is that suppression of the 4Hz oscillation is essential the model is to prove useful during closed loop simulation!

6.4.4 In search of trouble

This section deals with the isolation of the cause of the 4Hz gyro oscillation and its removal.

6.4.4.1 The cause

The following information and characteristics are known about the problem:

- A 4Hz gyro nutational oscillation with 0.3° amplitude results when the HIL simulation is activated.
- Before the gyro/reticle feedback comes into play, no sign of the oscillation is visible in the open loop system model. (The oscillation ceases during HIL implementation when the IR source is terminated.)
- The oscillation ceases momentarily during a discontinuous target position jump such as the step response in Figure 36.
- The oscillation's frequency, amplitude and phase vary only very slightly over time.

The following contradictions regarding the problem exist:

- The neural network is essentially a FIR system receiving 20ms of data sample history at its input. Sustaining a 4Hz oscillation with a period of 250ms through a FIR system having

³² The six central spokes are intended to aid in target resolution when scanning near the reticle origin, and also aids in passive secondary target and flare rejection.

only 20ms worth of signal history seems to be impossible. Less than a 12th of the signal period can be propagated through the control loop system.

- The pre-processing system's output employs a narrow 100Hz BPF that should not allow a 4Hz signal to pass.

The answer to the contradictions, along with the solution to the problem, lie in the pre-processing system. To indicate how the oscillation arises, sinusoidal inputs are injected into the existing pre-processing system. The sinusoids are of unity amplitude with frequencies of 1, 3, 4, 6, 10, 20, 40 and 60Hz respectively, with the pre-processing system's responses as indicated in Figure 39.

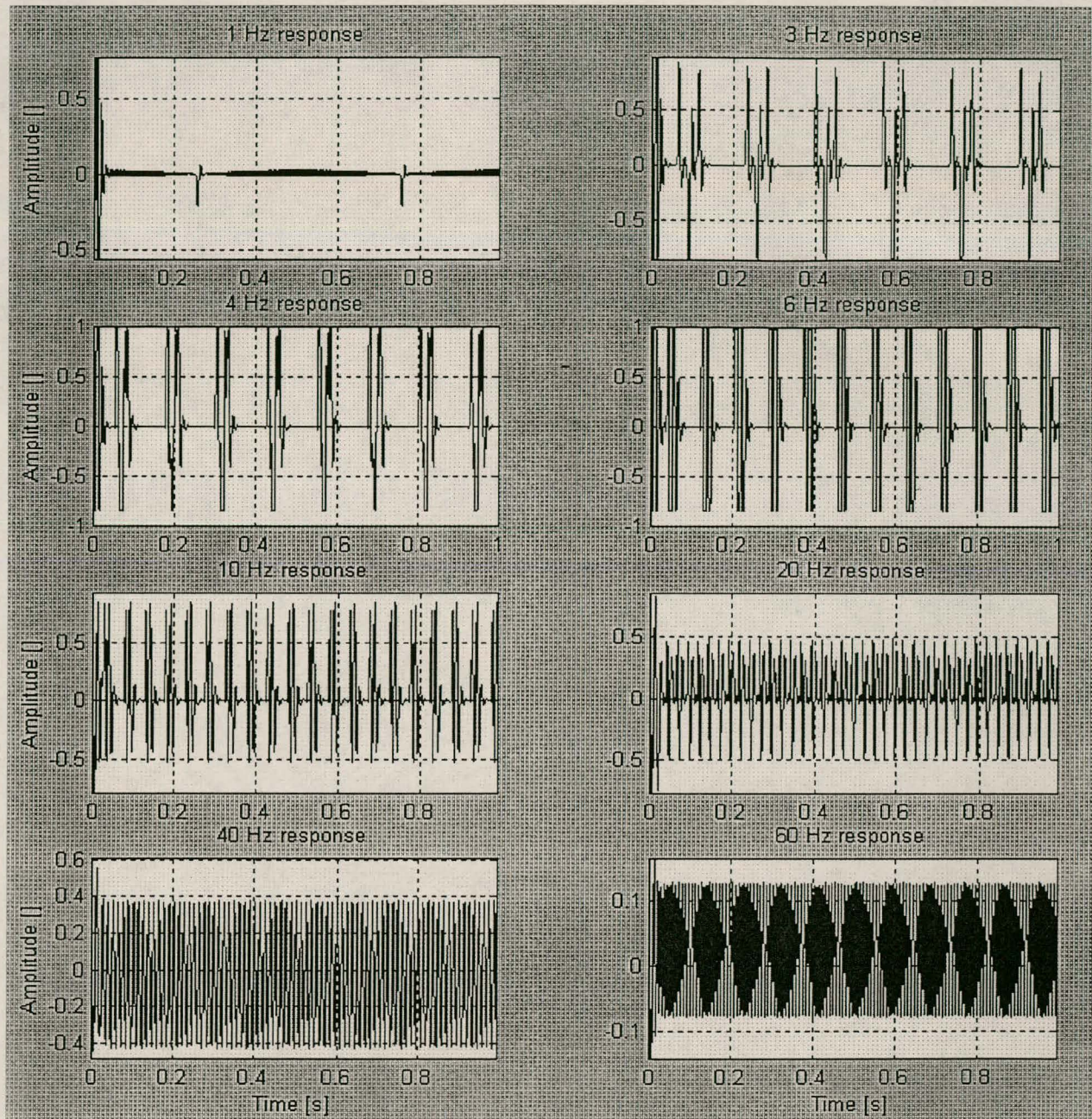


Figure 39 Open loop pre-processing model's responses sinusoid frequency inputs

The sinusoid injections of Figure 39 give some indications of interesting results:

- Frequency inputs of approximately 10Hz and below cause a high frequency ringing signal to result at the pre-processing output. The ringing re-occurs at twice the rate of the input sinusoid frequency.
- The main frequency component of the ringing remains in the vicinity of 50Hz.

- Frequency components above 20Hz are deformed, but are **partially** passed through the entire pre-processing structure.
- An AM component is visible in the 40 and 60Hz responses.
- Very low frequency inputs (e.g. 1Hz) have very little effect on the output. The ringing amplitude increases as the frequency increases up to approximately 6Hz. The amplitude of any output signals resulting from the sinusoidal inputs decreases continuously from 10Hz as the frequency is increased. (At about 100Hz the output reduces to an essentially wide-spectrumed noise signal with amplitude < 0.05 .)

It becomes clear that the HIL simulation's oscillatory results pose no true contradictions. An 50Hz ringing component may pass through the neural network, as a 50Hz signal has a 20ms period, which is identical to the network's input history sample window. The precession amplifier in the pre-processing system does not seem to suppress 50Hz sufficiently (around –12dB) in order to keep it from eventually having an effect on the gyroscope.

If, for example, the 50Hz ringing occurs 8 times a second, with the gyro spinning at 100Hz, the gyro is forced into a small 4Hz oscillation. The 4Hz oscillation causes 8 more ringing tones, as illustrated in Figure 39, and thereby could sustain the oscillation. The reasons for the sustained oscillation being at 4Hz, not at 6 or 8Hz, lies in the dynamics and bandwidth of the closed loop system. Time will not be spent on determining these reasons here. The cause of the ringing can be pinned down to the saturation of the preamplifier output and the demodulation system directly succeeding it. Time should preferably be spent on removing the ringing effects as far as possible, thereby removing the oscillatory HIL simulation response.

Why then did the ringing effects slip past during the modelling procedures? With the hardware injection system described in Chapter 4.1 being able to inject only saturated/limited input signals into the missile optics, a BPF was used to ensure that the uneven frequency harmonics added during saturation did not corrupt the essential frequency bands of the injected signal. In the process, low frequency information in the injected signal was lost. This implied that the eventual training data extracted from the missile did not contain sufficient low frequency response data, making it very difficult for any optimisation routine (both for neural network and pre-processing system) to judge what the low frequency response of the model should be. The importance of this low frequency data was known before the BP filtering was completed,

but the mistake slipped by undetected! An experiment similar to the one in Figure 39 should have been completed before HIL simulation was attempted.

Even though this does indeed represent errors in the modelling implementation, these errors would have slipped past if HIL simulation was not completed. The open loop data fit of the previous chapter did not indicate any possibility of any for these errors. This proves the value of HIL simulation, and qualifies the time spent on the subject. This proves particularly so if the error can be removed. Section 6.4.4.2 attempts to do just this!

6.4.4.2 The solution

The solution proves to be relatively simple:

Add low frequency response data to the training data, to ensure that the pre-processing and neural network systems are trained to handle low frequency data correctly.

The judgement can quite easily be made that low frequency input signals to the missile with no modulation content should have close to no effect at all on the missile's precession signal. We also know that the neural network is not the true cause of the oscillation. It was designed to allow certain low frequency signals to pass in order to reshape the model's precession output. In this section, focus will rather be placed on re-optimising the pre-processing system to eliminate ringing as far as possible. The neural network should, however, be re-trained if the HIL simulations are to be repeated. This should be done purely as an extra precautionary measure, assuring increased accuracy.

The implementation of the pre-processing re-optimisation is as follows:

5 seconds worth of 1, 3, 4, 6, 10, 20, 40 and 60Hz sinusoids were added to the training data inputs and their equivalent output responses here defined as 0V DC. After re-optimisation of the system, the results in Figure 40 are obtained. Figure 40 clearly illustrates that the ringing behaviour previously noted has been reduced considerably. Some low amplitude ringing is still visible surrounding 6Hz, and continuous wave outputs result at frequencies higher than 10Hz. Frequencies above 20Hz are still partially propagated by the system, but with lower output amplitude. The frequency area about 4Hz is especially noteworthy for its low amplitude output. It is the opinion of the author that the added low frequency rejection abilities of the pre-processing system should be sufficient to eliminate the oscillation encountered previously.

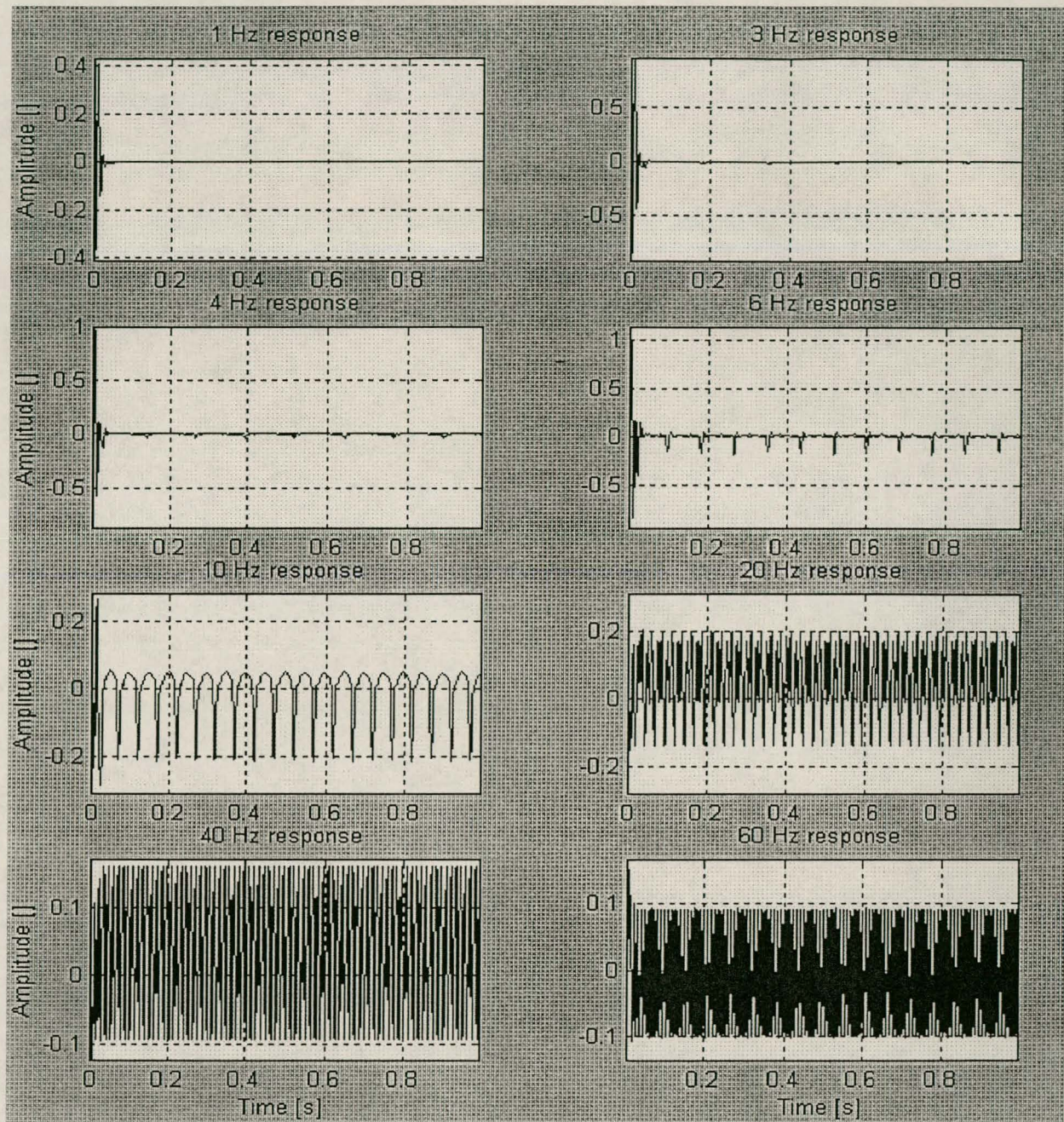


Figure 40 RE-OPTIMISED open loop pre-processing model's sinusoid input response

Some AM is still visible on the frequency responses above 40Hz, although their amplitudes are slightly smaller than before. The AM does not prove problematic at this stage, but it could prove worthwhile to examine the missile's response to these exact frequency inputs in order to come to a conclusion regarding the model accuracy at these frequencies.

After optimisation, it is also worthwhile to ensure that the pre-processing system's accuracy has not been adversely affected when confronted with target scenario inputs. Figure 41

provides an illustration of the same scenarios with which the original optimised pre-processing system was verified. When compared to Figure 26 and the discussion surrounding it, it can be seen that there is very little difference in accuracy between the newly optimised system and its predecessor, implying that the new pre-processing system may be used with confidence.

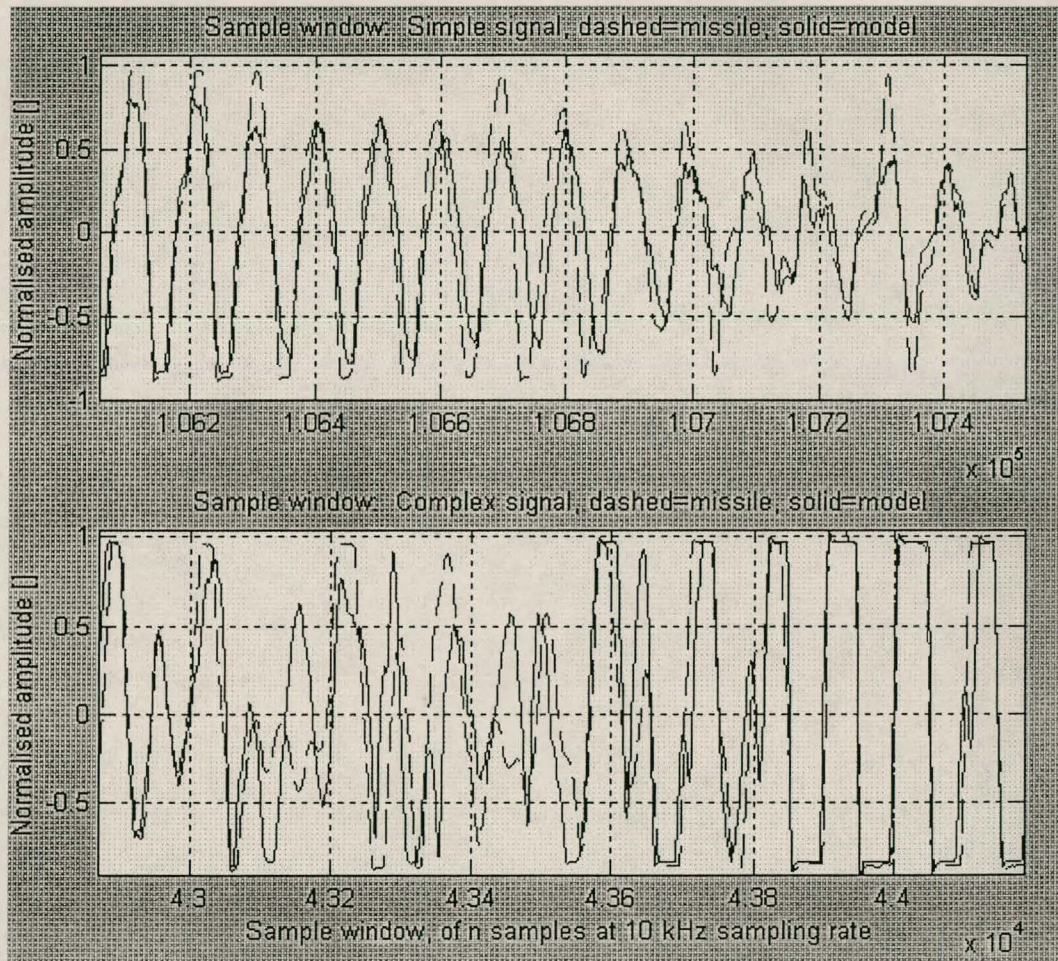


Figure 41 Typical re-optimised final Z-plane pre-processing results, ref. Figure 26

6.4.5 HIL validation of the re-optimised solution

Figure 41 indicates that the theoretical results of the re-optimisation have indeed improved the missile model's response and accuracy. Thus, a repeat of some of the HIL tests is required in order to prove conclusively that the 4Hz oscillation has indeed been removed.

A visit to Aerotek CSIR during February 1999 was earmarked for the repeated HIL test run. A set of rate tracking experiments was completed, during which the rate tracking ability of the model was validated. Close attention was paid to the possibility of the advent of any gyro

oscillation. Figure 42 illustrates the typical response of the missile's gyroscope during a rate tracking manoeuvre as effected by the model, compared to the true missile's response.

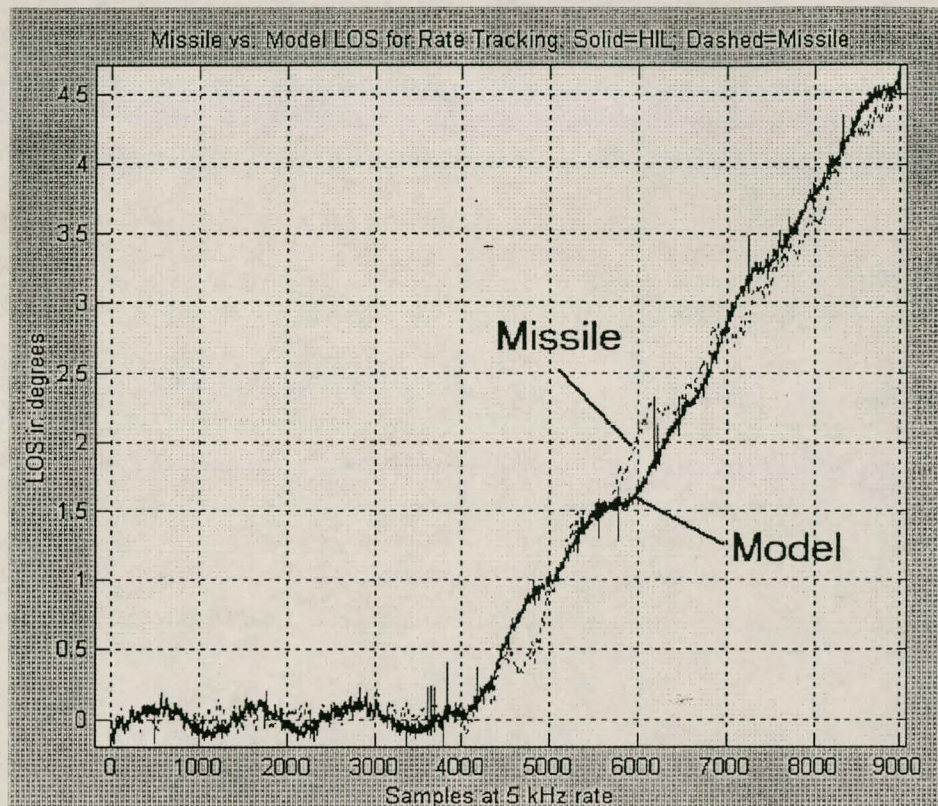


Figure 42 Re-optimised model's HIL rate tracking response

Figure 42 clearly illustrates a much reduced 4Hz oscillation, compared to the original response in Figure 38. **The significant reduction in the oscillation indicates that the steps taken to suppress it have indeed been successful!** It has not however, been completely eradicated, but when the missile's response to the same rate tracking manoeuvre is observed, a slight oscillation of the same frequency is also seen. This fact indicates that the stimulus exciting the 4Hz oscillation also excites the true missile system into slight oscillation. The responses of both the missile and the model are comparable in size and nature. This proves an extremely positive result. However, the question now arises as to why no oscillation is visible in the true missile response of Figure 38?

The reason for this is straightforward:

The rate tracking experiment of Figure 38 was completed at a lower tracking rate than that of Figure 42. Looking at the worm screw driven IR source mounted on the rate table it is clearly

visible that friction and/or a slightly bent worm screw causes the target to oscillate slightly at higher tracking rates. It is this oscillation that excites the missile's apparently "oscillatory" response in Figure 42. This indicates that both the missile and the model are susceptible to the 4Hz gyro oscillation, but that the model is still slightly more sensitive than the true missile. This is proved by the fact that the model maintains the oscillation with minimal or no driving input during, for example, stationary target tracking. In this case, the oscillation is of the same amplitude as in Figure 42. The true missile only maintains gyro oscillation under specific driving conditions, but results in an oscillation of roughly the same amplitude as the model's response.

The model therefore seems to be more accurate after the re-optimisation.

An additional piece of information was also uncovered during the course of re-optimisation:

The sensitivity of this specific missile's control system to high frequency bursts at 4Hz intervals might prove to be an effective countermeasure in the future!

CHAPTER 7

PROJECT EVALUATION AND CONCLUSIONS

7.1 Accomplishments

- An accurate neural network and deterministic processing hybrid model was developed of the tracking loop of the given missile, using rapid modelling techniques.
- A successful HIL simulation was completed during which the model's target tracking response was compared to the original missile's response.

7.1.1 Signal injection

- Various criteria were developed to assist in the choice of SID data to be injected into a missile, for the purposes of model and neural network optimisation.
- An IR battlefield simulator was used to generate simulated missile approach flight scenes, using radiometrically correct images of a C130 Hercules and an Oryx helicopter.
- A software model was developed of the missile optics and used to convert the flight scenes into electrical IR detector output signals.
- A hardware and software signal injection, and a recording system were developed for the system whereby the missile's response to the injection material could be captured.

7.1.2 Modelling

- Three main types of modelling structures were analysed regarding their accuracy and practicality.
- Various innovations were then added to the optimised hybrid modelling structure which was decided upon. These include: inter-module down-sampling techniques, partial Z-transformation, and re-optimisation.
- Modelling heuristics regarding neural networks and temporal signals were developed and summarised, in order to make an informed choice regarding the type, form and size of suitable artificial neural systems for this problem.
- Various structural characteristics/features of the control system were extracted from the missile by analysing the missile's outer structure and optics. These include the CSFM nature of target tracking, the FM carrier frequency, preamplifier bandwidth and

demodulation techniques used. These features were implemented and optimised within the pre-processing structure of the missile and used to determine the neural network architecture that is implemented.

- An accurate, real-time model of the target tracking loop was developed using Simuwin and SNNS software.

7.1.3 Model implementation and HIL

- The tracking loop model was hard-coded in C, including the pre-processing structure, the neural network and the A/D and D/A drivers, which made the model a stand-alone and practical entity that is able to replace the missile's target tracking electronics.
- A HIL simulation set-up was constructed, including the modelling software, missile gyroscope, gyro spin electronics, etc. in order to validate the model.
- The model's target tracking abilities were compared to those of the missile, delivering favourable results. The exception was a low amplitude, low frequency gyro mode that was shown to be existent within both the missile and the model, but slightly more pronounced within the model.

7.1.4 Gyroscopic modelling

- The missile's gyroscope was removed and the necessary sensing coils surrounding it were calibrated according to the gyro look angle during an operational test on a three axes rate table.
- An accurate gyro model was then developed by making use of AM precession signal injection into the gyro precession coils and Euler equation-based model optimisation.

7.2 Limitations

- HIL simulation is influenced by a low amplitude 4Hz gyro oscillation. The cause of the oscillation was isolated and the mode was suppressed. The mode is also existent within the true missile, but is still slightly more pronounced within the model.
- The modelling structure surely has limitations outside its training data scope, especially the neural network. Although the model functions well within its training boundaries, there exists no guarantee that the model would function correctly under very complex engagement scenarios.
- Although the modelling structure delivers accurate results, it should by no means be accepted that it is the best possible model for this missile's tracking loop. The nature of the

modelling techniques and the problem at hand could well mean that this is not the optimum rapid modelling model, but simply a sufficiently optimum model.

- The model was developed using rapid modelling techniques. This method has its advantages but also several disadvantages. Table 3 on page 35 provides a summary of the main advantages and disadvantages associated with this type of modelling.
- Cascading various optimised models invariably introduces accumulated modelling errors. The model consists of four separately optimised structures, with each following structure finding it very difficult (or impossible) to correct the errors made by previous structures. Such errors place additional strain on optimisation procedures. A neural network structure, such as the one used in the model, attempts to correct these errors and to add additional functionality. This will invariably limit its performance capabilities.
- The A/D and D/A sampling rate of 10kHz has some detrimental influence on the accuracy with which the high order z-transform structures are able to represent the pre-processing model when dealing with signals of 1 to 5kHz.

7.3 Possible improvements

- An improvement has already been made regarding the removal of a 4Hz gyro oscillation during the HIL test phase. Even after isolation and suppression of the mode, it is still more pronounced than its equivalent mode within the true missile, and further steps may be taken to ensure further suppression.
- Additional modelling structures may be considered when attempting rapid modelling of a system such as this. Neural network hybrids are not necessarily the best structures with which to solve the problem.
- Faster A/D and D/A converter as well as PC hardware may be used to increase the sampling rate used by the model. As certain frequency components within the system already approach the current Nyquist folding frequency of 5kHz, increasing the sampling rate will increase the model accuracy. Faster hardware could also allow additional processing to be completed within each sampling interval.
- The gyroscopic model may be added to the model to construct a fully software model of the tracking loop, when hardware processing speed limitations have been overcome³³. This

³³ Unless of course we are willing to wait an hour or so for 10 s of flight to be simulated using current non-real-time systems.

would require real-time scene generation, requiring feedback from the gyro model in order to create an accurate IR detector signal in real time.

- A mechanical and aerodynamic model of the missile may be added to the model in order to simulate the entire missile flight. This falls outside the scope of the current project, but the author constructed such a model in 1997 that may be linked to the tracking model. Restraints regarding current computing technology would unfortunately hinder such a resulting model from functioning in real-time.

7.4 Final remarks

All the goals of this project have been achieved. It should always be kept in mind that a model is exactly that: a model! All models have limitations, as does this one of the missile's tracking loop. What is important, however, is the pursuit of model accuracy, as far as is practically possible. It is believed that this relentless pursuit was diligently followed throughout the course of the project. The modelling structure and the criteria set down for the development and optimisation of such structures were kept as general as possible, while still including specific detail pertaining to this specific problem, whenever necessary. This was done specifically in order to assure that individuals or companies tackling similar problems could possibly make use of much of the general modelling, verification and signal injection criteria.

It can therefore be concluded that the problem of rapid modelling of a specific SAM's target tracking loop has been successfully completed.

REFERENCES

- [1] Aerotek CSIR, Various internal reports available on request from the IR Division, Defence Electronics Program, Aerotek, CSIR, PO Box 395, Pretoria, 0001, South Africa.
- [2] Agarwal, M., (1997) *"A systematic classification of neural-network-based control"*, IEEE Control Systems April 1997.
- [3] Bryson, A.E., (1994) *"Control of spacecraft and aircraft"*, Princeton University Press.
- [4] Chao, Z.W. et al, (1998) *"General analysis of FM reticles"*, Optical Engineering, Vol. 27, No. 6, June 1988.
- [5] Chao, Z.W. et al, (1998) *"Parameter analysis for FM reticle design"*, Optical Engineering, Vol. 27, No. 6, June 1988.
- [6] Chatfield, A.B., (1997) *"Fundamentals of high accuracy inertial navigation"*, Progress in astronautics and aeronautics, Vol. 174, AIAA.
- [7] Driggers, R.G. et al, (1991) *"Parameters of spinning FM reticles"*, Applied Optics, Vol. 30, No. 7, 1 March 1991.
- [8] Jones, T., (1997) *"Characterisation and modelling of a passive infrared guided surface to air missile"*, B.Eng final year thesis, University of Stellenbosch.
- [9] Kandel, A. et al, (1994) *"Fuzzy control systems"*, CRC Press.
- [10] Lin, C. et al, (1996) *"Neural fuzzy systems"*, Prentice Hall PTR.
- [11] Lippman, R.P., (1987) *"An introduction to computing with neural nets"*, IEEE ASSP magazine, April 1987.
- [12] Mehrotra, K. et al, (1997) *"A jerk model for tracking highly maneuvering targets"*, IEEE transactions on aerospace and electronic systems, Vol. 33, No. 4, October 1997.
- [13] Ogata, K., (1990) *"Modern control engineering"*, Prentice Hall.
- [14] Passino, K.M. et al, (1998) *"Fuzzy control"*, Addison-Wesley.
- [15] Phillips, C.L. et al, (1995) *"Digital control system analysis and design"*, Prentice Hall.

- [16] Proakis, J.G. et al, (1996) *"Digital signal processing principles, algorithms and applications"*, Prentice Hall.
- [17] Stremler, F.G., (1990) *"Introduction to communication systems"*, Addison-Wesley publishing company.
- [18] Zarchan, P., (1990) *"Tactical and strategic missile guidance"*, Progress in astronautics and aeronautics, Vol. 124, AIAA.
- [19] Zurada, J.M., (1992) *"Introduction to-artificial neural systems"*, West publishing company.

APPENDIX A: GYROSCOPIC MODELLING

This appendix covers the steps taken in order to model the missile's gyroscope. The exercise was completed as part of the philosophy that the hybrid-neural target tracking model will eventually be implemented as a complete software system. This would include a software gyroscope model. The software gyroscope model will displace the need for partial HIL set-ups when making use of the model. This is a summary of the modelling process, without the inclusion of detailed information regarding the test set-up. Aerotek, CSIR will receive a full report on this matter, but full experimental repeatability is not an issue in this thesis, as the exact set-ups and results are of a classified nature. Rather, the focus rather falls on some of the interesting results obtained.

A.1 Coil output vs. Look angle calibration

The gyroscope set-up, very much similar to that of the HIL set-up presented in section 6.2, is mounted vertically in the centre of the pitching axis of a rate table. The gyro spin control is activated and it is caged towards the centre of missile longitudinal axis. The gyro is then uncaged and the rate table is pitched at an angle. The respective amplitudes of the signals resulting from the gyro cage coils are then recorded in order to calibrate the cage coil output voltage with the gyro look angle. After each measurement, the gyro is once more caged and the table is returned to 0° pitch. Each pitch angle is repeated three times, with the average voltage of these runs being recorded. This is done since the gyro used in the experiment has been in use for quite some time, the likelihood being that its gimbal bearings are slightly worn out. This can sometimes cause the table pitching manoeuvre to influence the gyro orientation. For the same reason, it is essential to make the voltage measurement as soon as possible after the table has settled into a specified pitching angle, as the gyro look angle drifts slightly over time. The gyro is under no control³⁴ after being uncaged, as no orientational control electronics are included in the set-up. The gyro's inertial independence is fully relied upon when it is spun up to a rate of 100Hz.

³⁴ Other than spin regulation.

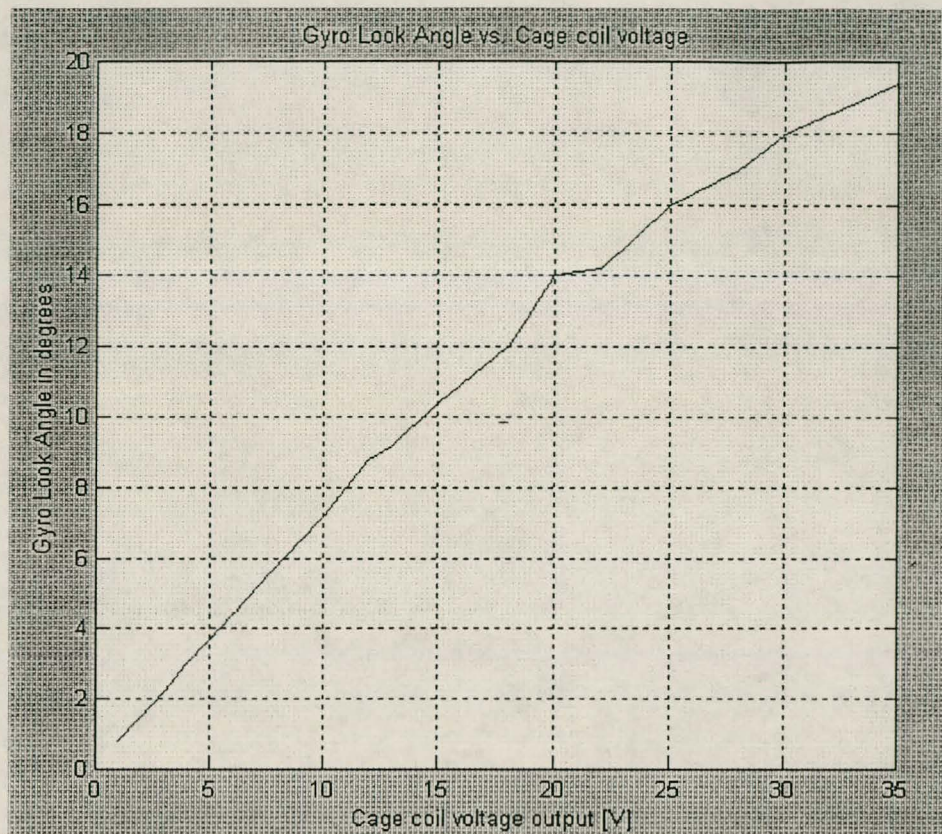


Figure 43 Gyro Look Angle vs. Cage coil voltage calibration

Figure 43 provides an illustration of some of the results obtained from the calibration experiment. The cage coil voltage output is a sinusoidal amplitude. The slight curve in the graph indicates that the gyro look angle/orientation within the coil is affecting the flux coupling between the coil and the gyro.

A.2 Gyro signal injection response measurements

A software program written in Pascal³⁵ is used to inject a signal onto the missile's precession coils and to record the gyro's response via the cage coils. A reference coil output provided the current gyro spin rate. The reference coil output was AM modulated with a signal between range 0.5 and 4Hz, and directly provided to the precession coils. One AM period of this type of driving resulted in the gyro being steered in a fixed plane. It starts at 0° look angle, moving outwards to an unspecified look angle in a straight line, after which it returns to its initial position. The gyro always moves in a straight line either outwards from or inwards to the

³⁵ See Appendix B.4 for the program.

central caging position. The precession voltage signal and the cage coils' outputs are recorded during each such an experiment.

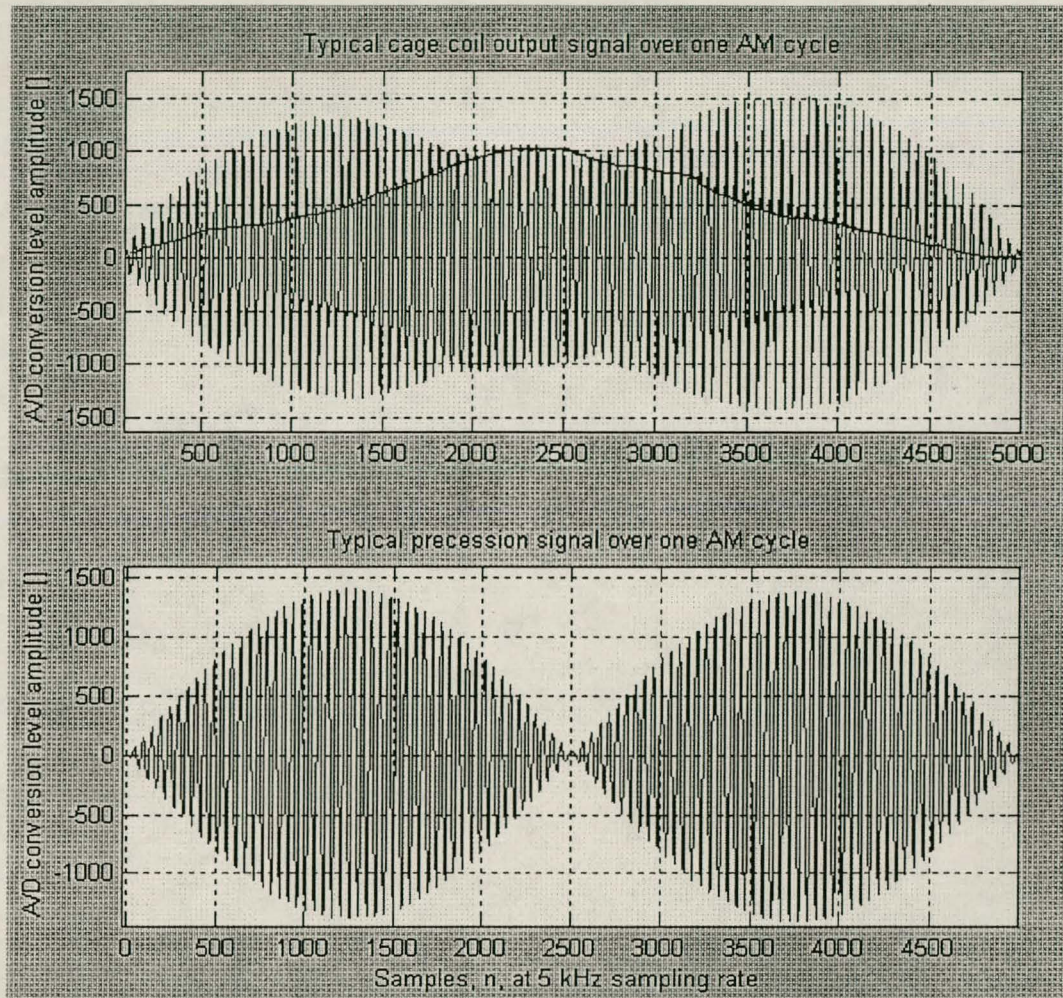


Figure 44 Typical precession signal and corresponding cage coil response

Figure 44 illustrates a typical precession signal input lasting one AM cycle, and the corresponding cage coil response. The precession signal consists of a signal with 100Hz frequency that is AM modulated by a 1Hz modulation frequency. The precession coils and the cage coils are wrapped in the same plane around the gyro, lying physically right on top of each other. This implies that the cage coils do not only pick up gyro look angle information, but are also influenced by any signal arriving at the precession coil, according to the principles of mutual inductance and the equation $V_c = L_M \frac{\partial I_p}{\partial t}$.

V_c represents the cage coil voltage component because of mutual inductance, L_m represents the mutual inductance between the coupling coils and I_p represents the current in the precession coil.

The solid line in the upper half of Figure 44 traces the portion of the cage coil signal envelope that represents the gyro look angle. The additional signal envelope is caused by mutual inductance between the precession control signal and the cage coil. The highest coupling voltage takes place at the highest $\frac{\partial I_p}{\partial t}$ rate on the precession signal. The coupling differs at varying gyro look angles, because of the influence the steel gyro itself has on the mutual inductance between the coils wrapped around it.

A.3 Model optimisation

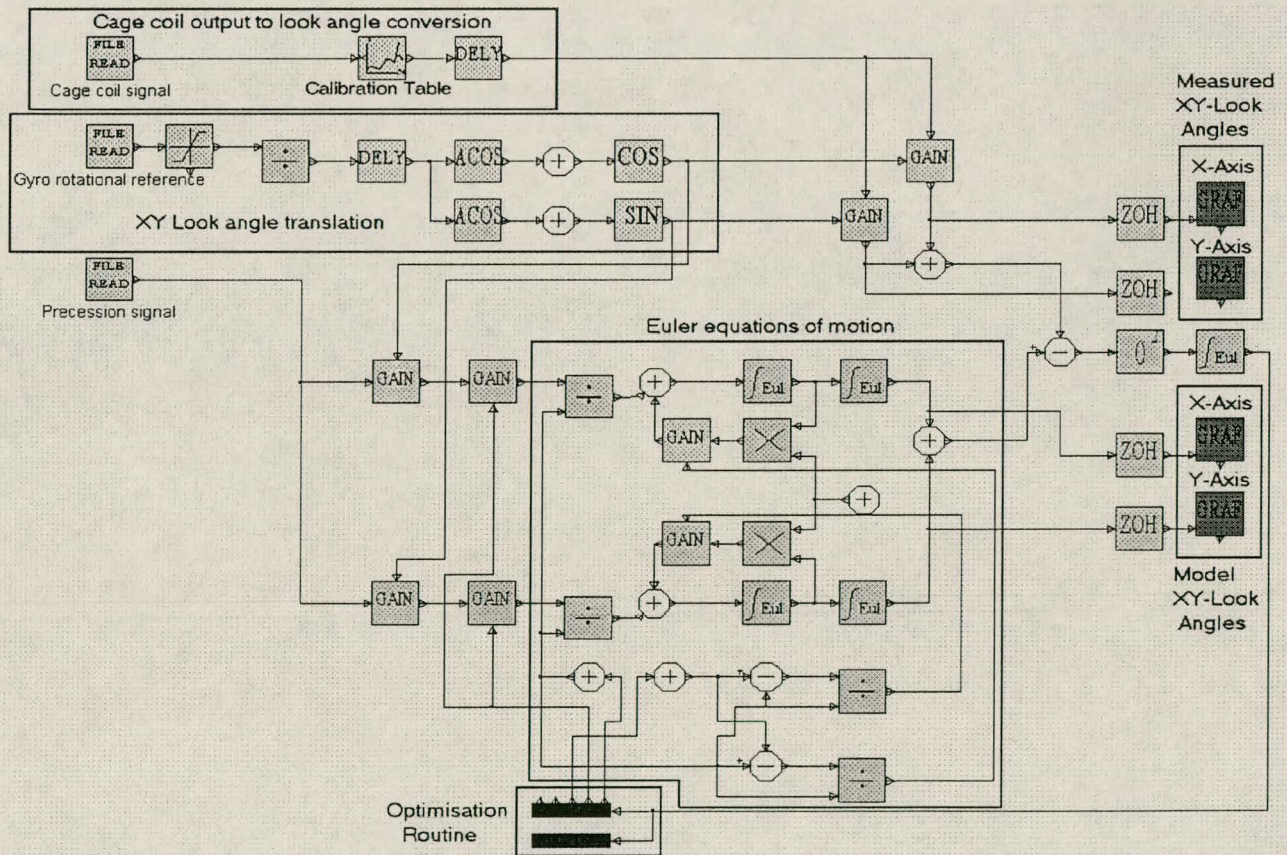


Figure 45 Gyro model optimisation structure

As useful parameters are extracted from such an optimisation, it was decided to fit an Euler equation model to the measured gyroscope response. The parameters requiring optimisation include the moments of inertia of the gyro about its gimbal, and a conversion factor that translates a voltage input to the precession coils into the applied torque/moment. Mutual inductance effects are removed from the cage coil measurements before it is used.

A calibration conversion table transforms the cage coil voltage into a radian look angle³⁶ in Figure 45. Translational mathematics transform the gyro reference signal into a gyro rotational angle and eventually a XY-axis (Polar to Cartesian conversion) gyro magnetic polarisation vector. This is done in order to translate the sinusoidal precession signal into a XY (Cartesian) force acting on the gyro, as the gyro is only N/S-magnetised in one direction. It is also used to provide a reference from which the gyro look angle can be converted from phase and amplitude information contained in a caging signal, into a XY look angle vector. The exact reference for the XY axes are not important, since it was never necessary for them to be measured, as long as the use of reference for the model and the measured data remains consequential.

Two delay blocks are visible in Figure 45. These compensate for the time it takes a 100Hz signal to pass through the A/D converter, before a comprehensive A/D converter model became available. However, the precession signal was not recorded through any A/D converter channel, and the delay was not needed. The delay does however also compensate for the ZOH lag experienced when D/A converting the precession signal.

The initial moments of inertia of the gyro were chosen to be equal to that of a gyro with well-known parameters and near equal size and weight to the one belonging to the missile. The X and Y axis moments of inertia are equal due to gyro symmetry. The remaining moment of inertia lies around the gyro spin axis.

None of the optimised gyro parameters differed more than 20% from the initial values, with the gyro model fitting the measured data to within a 0.16 degree error mean over one AM precession cycle. These results have proven to be extremely satisfactory.

³⁶ See Appendix C for more detail.

APPENDIX B: SOFTWARE CATALOGUE

B.1 Optics/Reticle emulation software (*Matlab*)

```
% Thomas Jones
% True InSb generation
% 06/02/1998
% Usage: [InSbSig,t //,RotReticle,SceneMatrix] =
InSbConde2(Time,TimeStep,Resolution,OnOff,ReticleStep,SceneStep)

function [InSbSig,t] =
InSbConde2(Time,TimeStep,Resolution,OnOff,ReticleStep,SceneStep)

%----- Constants -----
DefaultStep=100e-6;
DefaultTime=0.01;
DefaultRes=257;
DefaultSceneStep=10e-3;
DefaultReticleStep=10e-3;

GyroAngVel=100*2*pi;
BodyAngVel=14*2*pi;

SceneRotRadius=0.9; % 0.5 Degrees mirror offset
TotalFOW=3;
InstantFOW=2;
ScenePixRotRadius=(SceneRotRadius/TotalFOW)*Resolution/2;
%-----

%----- Init Variables-----
SceneCounter=0;
ReticleCounter=1;
InSbSig=[];
%-----

%----- Default Init -----
%--Error messages and auto init of inputs to functions
if isempty(TimeStep)
    TimeStep=DefaultStep;
    Warning=['WARNING: Default Time Step Set'];
    disp(Warning);
end;
if isempty(ReticleStep)
    ReticleStep=DefaultReticleStep;
    Warning=['WARNING: Default Reticle Intervals Set'];
    disp(Warning);
end;
if isempty(Time)
    Time=DefaultTime;
    Warning=['WARNING: Default Signal Duration Set'];
    disp(Warning);
end;
if isempty(Resolution)
    Resolution=DefaultRes;
    Warning=['WARNING: Default Reticle Resolution Set'];
    disp(Warning);
end;
if isempty(SceneStep)
    SceneStep=DefaultSceneStep;
    Warning=['WARNING: Default Scene Intervals Set'];
    disp(Warning);
```



```

end;
%-----

%----- Make Initial Reticle -----
disp('Making Inverse Reticle');
Reticle=GremRet2(0,Resolution);
%RotReticle=abs(Reticle-1); % Inverion of reticle
RotReticleSize=size(RotReticle);
XCeilMid=ceil(RotReticleSize(2)/2);
XFloorMid=floor(RotReticleSize(2)/2);
YCeilMid=ceil(RotReticleSize(1)/2);
YFloorMid=floor(RotReticleSize(1)/2);
%-----

%----- Generate Time Vector -----
t=[0:TimeStep:Time];
%-----

%----- Main Loop -----
disp('Making InSb Signal');
for n=[0:1:floor(Time/TimeStep)]
    %..... Scene Load Logic .....
    %Determines if new input flight scene slide is to be used (new scene
    %every 10 ms)
    if (n*TimeStep>=SceneCounter*SceneStep)
        disp('Loading New Scene');
        SceneCounter=SceneCounter+1;
        disp(SceneCounter);

        % Make Correct Filename to Load
        if (SceneCounter<10)
            SceneFileName=['reeks2_000'int2str(SceneCounter) '.out'];
        elseif (SceneCounter<100)
            SceneFileName=['reeks2_00'int2str(SceneCounter) '.out'];
        elseif (SceneCounter<1000)
            SceneFileName=['reeks2_0'int2str(SceneCounter) '.out'];
        end;
        % -----
        %Load new scene
        Scene=load (SceneFileName);
        Scene=uint8(Scene);
        SceneSize=size(Scene);
        %Determine scene centre
        SceneMidX=round(SceneSize(2)/2);
        SceneMidY=round(SceneSize(1)/2);
    end;
    %.....
    %..... Reticle Rotation Logic ...
    if (n*TimeStep>=ReticleCounter*ReticleStep)
        %-----
        disp('Rotating Reticle');
        ReticleCounter=ReticleCounter+1;
        RotReticle=imrotate(Reticle,-
(BodyAngVel*n*TimeStep*360/(2*pi)),'bilinear');
        %RotReticle=abs(RotReticle-1); % Inversion of reticle

        RotReticleSize=size(RotReticle);
        ReticleMidX=round(RotReticleSize(2)/2);
        ReticleMidY=round(RotReticleSize(1)/2);

        %Resize Rotated Image
        RotReticle=RotReticle(ReticleMidY-
floor(Resolution/2):ReticleMidY+floor(Resolution/2),...
        ReticleMidX-floor(Resolution/2):ReticleMidX+floor(Resolution/2));
        %.....

```

```

    RotReticSize=size(RotRetic);
    XCeilMid=ceil(RotReticSize(2)/2);
    XFloorMid=floor(RotReticSize(2)/2);
    YCeilMid=ceil(RotReticSize(1)/2);
    YFloorMid=floor(RotReticSize(1)/2);
end;
%.....
%..... Retic/Scene Scan Rotation
%Offset due to angled front mirror of telescope (CSFM)
XPixOffset=round(ScenePixRotRadius*sin(GyroAngVel*n*TimeStep));
YPixOffset=round(ScenePixRotRadius*cos(GyroAngVel*n*TimeStep));
%.....
%..... Cut Out Scene View .....
XWindowMin=SceneMidX+XPixOffset-XCeilMid;
XWindowMax=SceneMidX+XPixOffset+XFloorMid;
YWindowMin=SceneMidY+YPixOffset-YCeilMid;
YWindowMax=SceneMidY+YPixOffset+YFloorMid;
SceneMatrix=Scene(YWindowMin:YWindowMax-1,XWindowMin:XWindowMax-1);
%.....
%..... Retic/Scene Superpos ...
TransRetic=RotRetic.*double(SceneMatrix);
InSbSig(n+1)=sum(sum(TransRetic));
% Display image of scene and reticle overlap if required
if (OnOff>0)
    %contrst=contrast(TransRetic,50);
    imshow(uint8(TransRetic));
    colormap(prism);
    colorbar;
    pause(0.1);
end;
%.....
end;
%-----

```

B.2 Injection/Recording software (*Pascal*)

```

program FINAL2;
uses
    DOS, GRAPH, CRT;
const
    PERIODE = 1;
    ADC     = $360;           {Addresses of ADDAS card}
    ADChi   = $362;
    DAC0    = $364;
    DAC1    = $366;
    {filename = 'd:\teststep.dat';}
    {writefile = 'd:\ADOUT.dat';}

type
    lintfiletype = file of longint;
    wordfiletype = file of word;

var
    VLAG      : boolean;
    ADDATA    : array[0..3] of integer;
    VEKTOR    : pointer;
    segarray, offarray, segarrayw, offarrayw : array [0..7] of word;
    nn, mm, test : word;
    datafile   : lintfiletype;
    outfile    : wordfiletype;
    p1, p2, p3, p4, p5, p6, p7, p8, w1, w2, w3, w4, w5, w6, w7, w8 : ^byte;
    filename, writefile : string;

```



```

procedure AtoD;
begin
  inline(
    $BA/$60/$03/      {mov dx,ADC}
    $EC/              {in al,dx}                {Dummy Read}
    $EE/              {out dx,al}                {Start conversion}
    $B9/$00/$01/      {mov cx,100H}
    $BA/$62/$03/      {mov dx,ADCHi}
    {@1:}
    $EE/              {out dx,al}                {Clock interrupt line in
latch}
    $EC/              {in al,dx}                {Poll ADC interrupt}
    $24/$80/          {and al,80H}
    $E0/$FA/          {loopnz @1}
    $BF/ADDATA/       {mov di,offset ADDATA}
    $B9/$04/$00/      {mov cx,1}
    {@2:}
    $BA/$60/$03/      {mov dx,ADC}
    $EC/              {in al,dx}                {Read low byte}
    $88/$C3/          {mov bl,al}
    $BA/$62/$03/      {mov dx,ADCHi}
    $EC/              {in al,dx}                {Read high byte}
    $24/$0F/          {and al,0FH}
    $88/$C7/          {mov bh,al}
    $24/$08/          {and al,08H}
    $74/$04/          {jz @3}
    $81/$EB/$00/$10/  {sub bx,1000H}
    {@3:}
    $89/$1D/          {mov [di],bx}                {Store channel A/D data}
    $47/              {inc di}
    $47/              {inc di}
    $E2/$E4);        {loop @2}
end; {AtoD}

```

```

procedure DtoA(K,SS : longint);
var
  S : integer;
begin
  S:=integer(SS);
  if S > 2047 then S := 2047;                {Card DA resolution limit check}
  if S < -2048 then S := -2048;
  S := (S + 2048) shl 4;
  if K = 0 then begin                        {DA Port selection}
    port[DAC0+1] := hi(S);
    port[DAC0] := lo(S);
  end
  else begin
    port[DAC1+1] := hi(S);
    port[DAC1] := lo(S);
  end;
end; {DtoA}

```

```

procedure KLOK; assembler; {Sets VLAG when interrupt vectors to KLOK}
asm
  push  ax
  push  ds
  mov   ax, seg @DATA
  mov   ds, ax
  mov   VLAG, 1
  mov   al, 20H
  out   20H, al
  pop   ds

```

```

    pop    ax
    iret
end;

```

```

procedure INIT;           {Initialisation procedure}
var
    II : byte;
begin
    getintvec($08,VEKTOR); {save old interrupt vector}
    port[$43]:=$36;        {ready onboard int clock to be set}
    port[$40]:=lo(119);    {50ms=59659, 20ms=23864}
    port[$40]:=hi(119);
    setintvec($08,@KLOK);  {set new vector}
end;

```

```

procedure STOP;           {termination procedure}
begin
    port[$43]:=$36;
    port[$40]:=0;
    port[$40]:=0;
    setintvec($08,VEKTOR);
    DtoA(0,0);
end;

```

```

{HOOFFPROGRAM}
begin
    write('DA Filename:');
    readln(filename);

    write('AD Filename:');
    readln(writefile);

    {delay(1000);}

    writeln('>> Allocating DA Memory <<');
    getmem(p1,41500);
    getmem(p2,41500);
    getmem(p3,41500);
    getmem(p4,41500);
    getmem(p5,41500);
    getmem(p6,41500);
    getmem(p7,41500);
    getmem(p8,41500);

    segarray[0]:=seg(p1^);
    segarray[1]:=seg(p2^);
    segarray[2]:=seg(p3^);
    segarray[3]:=seg(p4^);
    segarray[4]:=seg(p5^);
    segarray[5]:=seg(p6^);
    segarray[6]:=seg(p7^);
    segarray[7]:=seg(p8^);

    offarray[0]:=ofs(p1^);
    offarray[1]:=ofs(p2^);
    offarray[2]:=ofs(p3^);
    offarray[3]:=ofs(p4^);
    offarray[4]:=ofs(p5^);
    offarray[5]:=ofs(p6^);
    offarray[6]:=ofs(p7^);
    offarray[7]:=ofs(p8^);

    writeln('>> Allocating AD Memory <<');

```



```

getmem(w1,20750);
getmem(w2,20750);
getmem(w3,20750);
getmem(w4,20750);
getmem(w5,20750);
getmem(w6,20750);
getmem(w7,20750);
getmem(w8,20750);

segarrayw[0]:=seg(w1^);
segarrayw[1]:=seg(w2^);
segarrayw[2]:=seg(w3^);
segarrayw[3]:=seg(w4^);
segarrayw[4]:=seg(w5^);
segarrayw[5]:=seg(w6^);
segarrayw[6]:=seg(w7^);
segarrayw[7]:=seg(w8^);

offarrayw[0]:=ofs(w1^);
offarrayw[1]:=ofs(w2^);
offarrayw[2]:=ofs(w3^);
offarrayw[3]:=ofs(w4^);
offarrayw[4]:=ofs(w5^);
offarrayw[5]:=ofs(w6^);
offarrayw[6]:=ofs(w7^);
offarrayw[7]:=ofs(w8^);

writeln('>> Reading DA File <<');
assign(outfile,writefile);
rewrite(outfile);
assign(datafile,filename);
reset(datafile);
for nn:=0 to 7 do
begin
write('          Segment: ');
write(nn+1);
write(' ');
writeln(' ');
{-----Read entire DA file-----}
for mm:=0 to 10374 do
begin
read(datafile,meml[segarray[nn]:(offarray[nn]+4*mm)]);
end;
end;
{-----}

writeln('>> Init IRQ 8 <<');
INIT;
writeln('>> Start DA Conversion <<');

for mm:=0 to 7 do
begin
for nn:=0 to 10374 do
begin
repeat until VLAG; {Wag vir monster oomblik}
VLAG:=false; {wait for interrupt}
AtoD; {read AD data}
DtoA(0,meml[segarray[mm]:(offarray[mm]+4*nn)]); {Write DA data, -2048
to +2047 = -10 to +10 Volt}
memw[segarrayw[mm]:(offarrayw[mm]+2*nn)]:=word(ADDATA[0]+32768); {store
AD data in memory}
end;
end;
writeln('>> DA Conversion Completed <<');

```

```

writeln('>> Freeing DA Memory <<');

freemem(p1,41500);
freemem(p2,41500);
freemem(p3,41500);
freemem(p4,41500);
freemem(p5,41500);
freemem(p6,41500);
freemem(p7,41500);
freemem(p8,41500);
close(datafile);

writeln('>> Saving AD Data <<');
for nn:=0 to 7 do
begin
    write('        Save Segment: ');
    write(nn+1);
    write(' ');
    writeln(' ');
    for mm:=0 to 10374 do
    begin
        write(outfile,memw[segarrayw[nn]:(offarrayw[nn]+2*mm)]);
    end;
end;
writeln('>> Freeing AD Memory <<');
freemem(w1,20750);
freemem(w2,20750);
freemem(w3,20750);
freemem(w4,20750);
freemem(w5,20750);
freemem(w6,20750);
freemem(w7,20750);
freemem(w8,20750);
close(outfile);

STOP;
writeln('>> All Stop Halt etc.... <<');

end. {FINAL2}

```

B.3 HIL Simulation software (GCC)

Main simulation file: SNNS20d.c

```

#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include "timer.h"

#define ADC 0x360
#define ADCHi 0x362
#define DAC0 0x364
#define DAC1 0x366
#define DAScale 2000 //Scaling factors
#define ADScale 1

#define ON 1
#define OFF 0

char temp[5];
extern volatile int VLAG;
int ADDATA[4];
int INSET;

```



```

int UITSET;
float PROCHISTORY[191];
float *netinput, netoutput, PREPROCOUT, PREPROCOUT2;
int i,f,ti,numinputs,ADSAMP;
long unsigned int n;
//byte dummy1,dummy2;
unsigned int dummy1,dummy2,d3;
//word INWORD;
unsigned int INWORD;
float
x1,x1p1,x2,x2p1,x3,x3p1,x4,x4p1,x5,x5p1,x6,x6p1,x7,x7p1,INSET2,INSET1,y1;

void AtoD()          //AD conversion procedure
{
    //mov dx,ADC
    //in al,dx          // Dummy Read
    dummy1=inportb(ADC);
    //out dx,al          // Start conversion
    outportb(ADC,dummy1);
    dummy1=0x80;
    while((ADSAMP<0x100)&&(dummy1&0x80!=0))
    {
        //mov cx,100H
        //mov dx,ADCHi
        //@1:
        //out dx,al          // Clock interrupt line in latch
        outportb(ADCHi,dummy1);
        //in al,dx          // Poll ADC interrupt
        dummy1=inportb(ADCHi);
        //and al,80H
        //loopnz @1
    }

    //mov di,offset ADDATA
    //mov cx,4
    //@2:
    //mov dx,ADC
    //in al,dx          // Read low byte
    dummy2=inportb(ADC);
    //mov bl,al
    dummy1=dummy2;
    //mov dx,ADCHi
    //in al,dx          // Read high byte
    dummy2=inportb(ADCHi);
    //and al,0FH
    dummy2=dummy2&0x0F;
    //mov bh,al
    INWORD=(dummy2*256)+dummy1;
    //and al,08H
    if (dummy2&0x08) INWORD=INWORD-0x1000;
    //jz @3
    //sub bx,1000H
    //@3:
    //mov [di],bx          // Store channel A/D data
    INSET=(INWORD);          // OMSETTING?
    //inc di
    //inc di
    //loop @2
}

void DtoA(int K, int S) //DA conversion procedure
{
    if (S>2047)
        S=2047;

```

```

if (S<-2048)
    S=-2048;
S=(S+2048)*16; //Shift left 4
//outpw(DAC0,S);
outportb(DAC0+1,S/256);
outportb(DAC0,S%256);
}

void INIT()          //Initialisation procedure
{
    outportb(0x43,0x36); //Set clock
    outportb(0x40,0x77);
    outportb(0x40,0);
    for (d3=0;d3<4;d3++) ADDATA[d3]=0;
    for (d3=0;d3<191;d3++) PROCHISTORY[d3]=0;
    INSET=0;
    UITSET=0;
    x1=0;
    x2=0;
    x3=0;
    x4=0;
    x5=0;
    x6=0;
    x7=0;
    y1=0;
    INSET2=0;
    VLAG=0;
}

void STOP()
{
    outportb(0x43,0x36);
    outportb(0x40,0);
    outportb(0x40,0);
    DtoA(0,0);
}

void main(void)
{
    INIT();          //INIT ADAS Kaart
    printf("Interrupt Shutdown, EXIT in 20 seconds, press <ENTER>.\n");
    gets(temp);
    if (timer_init())          //INIT TIMER Interrupt
    {
        printf("DPMI error\n");
        exit(-1);
    }
    numinputs = 20;
    netinput = (float *) calloc(numinputs + 1, sizeof(float));
    i=0;
    for (n=0;n<200000;n++)
    {
        while (VLAG==0);      //WAIT FOR INTERRUPT
        AtoD();
        DtoA(0,UITSET);
        //printf("%d",n);
        VLAG=0;
        //PREPROC(VOID);

        // -----PRE-PROCESSING STRUCTURES-SEE SIMUWIN BLOCK DIAGRAMS
        INSET1=INSET*ADScale;
        INSET1=INSET1*5.0;
        if (INSET1>10) INSET1=10.0;
    }
}

```



```

    if (INSET1<-10) INSET1=-10.0;

    x1p1=1.04231*x1+x2-0.00116957*INSET1;
    x2p1=-0.416725*x1+0.00217147*INSET1;
    y1=x1-0.000752926*INSET1;

    if (y1<0) y1=y1*(-1.0);

    INSET2=-6.73618*(y1);

    x3p1=4.2541049195*x3-7.108247822*x4+5.79331286*x5-
2.27857799*x6+3.3940162e-1*x7+INSET2;
    x4p1=x3;
    x5p1=x4;
    x6p1=x5;
    x7p1=x6;

    PREPROCOUT=1.20850275e-3*x3+1.331275521e-3*x4-5.58289478e-
3*x5+2.3379521e-3*x6+7.05164423e-4*x7;

    x7=x7p1;
    x6=x6p1;
    x5=x5p1;
    x4=x4p1;
    x3=x3p1;
    x2=x2p1;
    x1=x1p1;

    // N-NETWORK INPUT MANAGEMENT

    //extra PREPROC
    PREPROCOUT2=PREPROCOUT*80.0;
    if (PREPROCOUT2>0.98) PREPROCOUT2=0.98;
    if (PREPROCOUT2<-0.84) PREPROCOUT2=-0.84;
    // End of extra

    if (i>190) i=0;
    PROCHISTORY[i]=PREPROCOUT2;
    f=i;
    for(ti=0;ti<20;ti++)
    {
        if(f-ti*10<0) f=f+191;
        netinput[ti]=PROCHISTORY[f-ti*10];
    }
    i++;
    SNNS20d(netinput, &netoutput, 0); //NETWORK CALL
    UITSET=(0.1)*netoutput*DAScale; //MOD on 14/10/98 at CSIR
}
----- //End of main for loop, n-counter
/* Uninstall the handler: */
    STOP();
    timer_close();
}

```

Protected mode timer interrupt header: Timer.h

/* Timer handling functions for DJGPP v2.00, by Thomas Jones */

```

#ifndef _TIMER_H
#define _TIMER_H

#ifdef __cplusplus
extern "C" {

```

```

#endif

extern int timer_init(void);
extern void timer_close(void);

#ifdef __cplusplus
}
#endif

#endif /* _TIMER_H */

```

Protected mode timer interrupt: Timer.s

Timer Interrupt, by T Jones

```

        .file "timer.s"

        .extern __djgpp_base_address
        .extern __djgpp_ds_alias
        .extern __djgpp_dos_sel

# public functions and variables:

        .global _timer_map
        .global _timer_init
        .global _timer_close
        .global _VLAG
        .text

        .align      4

locking_region_start:

_VLAG:      .long    0
old_vector:
old_vector_ofs: .long 0
old_vector_sel: .word 0
chain_flag: .long    1

        .align      4

handler_procedure:

# Initial pushes
        pushl %eax
        pushl %edx
        pushw %ds

#
# Load DS with our data selector
#
        movw %cs:__djgpp_ds_alias, %ds

# Set VLAG and Acknowledge interrupt

        movl    $1, _VLAG
        movb    $0x20, %al
        outb    %al, $0x20

# NON-CHAINING PART
        popw    %ds
        popl    %edx
        popl    %eax
        sti

```



```

        iret

        .align      4

handler_chain:  popw    %ds
#               popl    %edx
#               popl    %eax
#               ljmp    %cs:(old_vector)

locking_region_end:

        .align      4
_timer_init:

#
# int timer_init(void);
#
# Initializes the timer handler and hooks the timer interrupt.
# Returns -1 on failure, zero on success
#
        pushl %esi
        pushl %edi
        pushl %ebx
#
# First, we need to lock the handler and memory it touches, so
# it doesn't get swapped out to disk.
#

        leal  locking_region_start, %ecx
        leal  locking_region_end, %edi
        subl  %ecx, %edi
        addl  __djgpp_base_address, %ecx
        shldl $16, %ecx, %ebx      # ecx -> bx:cx
        shldl $16, %edi, %esi      # edi -> si:di
        movw  $0x0600, %ax         # lock linear region
        int   $0x31
        jc    init_error

#
# Now we need to save the old interrupt vector, so we can restore
# it later and also to know where to jump if chaining.
#
        movw  $0x0204, %ax         # get pm int vector
        movb  $0x08, %bl
        int   $0x31
        movw  %cx, old_vector_sel
        movl  %edx, old_vector_ofs

# Set the interrupt vector to point to our handler.
#
        movw  %cs, %cx
        leal  handler_procedure, %edx
        movb  $0x08, %bl
        movw  $0x0205, %ax         # set pm int vector
        int   $0x31

#*
#* Actually we would have to unlock the locked region on failure
#* here. But since most programs would exit with an error message
#* in such case, there's no need to worry.
#*

init_error:

```

```

#
# This sets EAX to -1 if CF is set and to 0 otherwise
#
        movl    $0, %eax
        sbbl    $0, %eax

        popl    %ebx
        popl    %edi
        popl    %esi
        ret

        .align 4
_timer_close:

#
# void timer_close(void);
#
# Shuts the timer handler down.
#
        cli
        pushl   %esi
        pushl   %edi
        pushl   %ebx

#
# Unlock the region we locked at initialization
#
        leal    locking_region_start, %ecx
        leal    locking_region_end, %edi
        subl    %ecx, %edi
        addl    __djgpp_base_address, %ecx
        shldl   $16, %ecx, %ebx
        shldl   $16, %edi, %esi
        movw    $0x0601, %ax          # unlock linear region
        int     $0x31

#
# Restore the interrupt vector to its previous value
#
        movw    old_vector_sel, %cx
        movl    old_vector_ofs, %edx
        movb    $0x08, %bl
        movw    $0x0205, %ax          # set pm int vector
        int     $0x31

        popl    %ebx
        popl    %edi
        popl    %esi
        sti
        ret

#
# void timer_chain(int toggle);
#
        .align    4

```

Neural network function header: SNNS20d.h (GEN. BY SNNS2C)

```

/*****
SNNS20d.h
-----
generated at Wed Oct  7 17:33:57 1998
by snns2c ( Bernward Kett 1995 )
*****/

```



```
extern int SNNS20d(float *in, float *out, int init);
```

```
static struct {
    int NoOfInput;    /* Number of Input Units */
    int NoOfOutput;   /* Number of Output Units */
    int(* propFunc)(float *, float*, int);
} SNNS20dREC = {20,1,SNNS20d};
```

Neural network function: SNNS20d.c (GEN. BY SNNS2C)

```
/******
SNNS20d.c
```

```
-----
generated at Wed Oct  7 17:33:57 1998
by snns2c ( Bernward Kett 1995 )
```

```
*****/
```

```
#include <math.h>
```

```
#define Act_Logistic(sum, bias) ( (sum+bias<10000.0) ? ( 1.0/(1.0 + exp(-
sum-bias) ) ) : 0.0 )
#define Act_Tanh_Xdiv2(sum, bias) ( tanh( (sum + bias) / 2) )
#define NULL (void *)0
```

```
typedef struct UT {
    float act;          /* Activation */
    float Bias;         /* Bias of the Unit */
    int NoOfSources;    /* Number of predecessor units */
    struct UT **sources; /* predecessor units */
    float *weights;     /* weights from predecessor units */
} UnitType, *pUnit;
```

```
/* Forward Declaration for all unit types */
```

```
static UnitType Units[42];
```

```
/* Sources definition section */
```

```
static pUnit Sources[] = {
```

```
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units +
7, Units + 8, Units + 9, Units + 10,
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 17, Units + 18, Units + 19, Units + 20,
```

```
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units +
7, Units + 8, Units + 9, Units + 10,
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 17, Units + 18, Units + 19, Units + 20,
```

```
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units +
7, Units + 8, Units + 9, Units + 10,
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 17, Units + 18, Units + 19, Units + 20,
```

```
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units +
7, Units + 8, Units + 9, Units + 10,
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 17, Units + 18, Units + 19, Units + 20,
```

```
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units +
7, Units + 8, Units + 9, Units + 10,
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 17, Units + 18, Units + 19, Units + 20,
```

```
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units +
7, Units + 8, Units + 9, Units + 10,
```



```
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units +
7, Units + 8, Units + 9, Units + 10,
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 17, Units + 18, Units + 19, Units + 20,
```

```
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units +
7, Units + 8, Units + 9, Units + 10,
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 17, Units + 18, Units + 19, Units + 20,
```

```
Units + 21, Units + 22, Units + 23, Units + 24, Units + 25, Units + 26,
Units + 27, Units + 28, Units + 29, Units + 30,
Units + 31, Units + 32, Units + 33, Units + 34, Units + 35, Units + 36,
Units + 37, Units + 38, Units + 39, Units + 40,
```

```
};
```

```
/* Weights definition section */
```

```
static float Weights[] = {
-1.732010, -1.034850, 2.779560, 1.288120, -0.789480, 0.119180, -0.339650,
1.226420, 2.571670, 0.524560,
-0.858580, -1.129960, 1.586290, -0.109740, 1.164480, -2.977210, -0.429000,
0.013080, -0.549050, -0.324790,

-0.838320, -4.988950, -1.717450, 4.962960, -2.939940, -1.196870, 2.293200,
0.425350, 5.524830, 3.979860,
0.260740, -1.911710, 0.410710, 8.828360, -4.671520, -1.706710, 0.758500,
2.228550, 8.186320, 0.655250,

2.418610, 3.940780, 1.954750, -0.699660, 3.306490, 4.553990, -0.541740, -
0.651740, 2.316060, 0.313090,
-6.766360, -1.880740, 1.605640, 1.134030, -2.792930, -4.800660, 3.881740,
6.065240, -2.298650, -4.028870,

1.160420, -2.232740, -2.454970, -2.424490, -2.380210, -1.454700, -0.109640,
-1.295170, -0.149390, 5.962570,
-0.818830, -0.603420, -3.271100, 5.472200, 3.911460, 3.406660, 0.207310, -
3.936210, 8.295970, 0.020720,

-0.133430, -0.139200, 3.197840, 1.629810, 2.962550, -1.357940, 1.921440,
0.259250, -2.626340, 1.361610,
0.065070, 1.032800, -1.759470, -3.152340, 0.305970, 2.392160, 0.993700, -
3.716940, -2.610430, -3.220150,

-1.310320, -0.069990, 2.832860, 2.792310, -0.985850, 0.613540, -0.287920,
3.380980, 1.116770, -0.071020,
-1.625120, -0.278550, 0.108040, 0.045540, -0.942740, -6.382420, -3.926720,
3.316980, 2.298630, 0.440600,

-0.376160, -1.947850, -0.837010, 1.196900, 2.854690, 1.635210, -1.370070, -
2.180020, -2.113930, 0.263760,
2.328760, -1.437850, 0.672320, 2.391140, 5.019380, 1.223760, -1.087650,
3.888770, 6.972840, 5.841820,

0.212850, -0.364570, 0.283050, 0.064250, -2.212010, -1.023970, 0.708000,
1.558340, 0.094350, -3.256860,
5.525740, 0.216270, -0.727870, -0.154900, 0.589400, -0.003910, -0.033570, -
0.961950, -0.656060, 1.897130,

1.265210, 0.136570, -2.723400, 3.756650, 1.126130, -0.781990, -1.258970, -
0.572360, -1.363490, 0.080710,
1.920970, -2.643280, -4.498650, 3.210200, 2.584040, -5.212150, -1.907610, -
1.972760, 4.244390, -1.998860,
```

```
-1.028170, -0.299550, 0.598630, 1.284780, 0.849930, -0.045440, -1.027960,  
2.549530, 0.889800, -1.138640,  
-1.625820, 0.208370, 1.323700, 0.888870, -0.186770, -1.476250, -1.683910,  
0.845810, 0.478980, -3.897490,  
  
0.491220, -0.244830, 1.110920, 3.362550, 3.392220, 1.007850, -0.597920,  
1.394040, -1.008850, -1.087750,  
0.965110, 0.754470, 0.481430, -0.993100, -2.325110, 0.388240, -0.294780, -  
0.399620, 1.864400, 0.687770,  
  
-18.455099, -4.090980, -7.116400, -2.189800, -15.639450, 11.285190, -  
10.384540, -6.912420, 0.222020, -3.218870,  
-11.701960, 0.009880, -1.265410, 4.659790, 4.002070, -3.741130, 3.362590,  
10.166130, 0.150090, -5.670910,  
  
-0.316250, 1.068620, -0.350220, -0.576780, -2.651460, -0.065930, 0.905190,  
1.237080, 1.924390, -0.517360,  
-0.338930, 0.884880, 1.173170, 2.741050, -3.121690, -1.027530, 1.004470,  
1.621200, 0.197200, -0.738820,  
  
3.189400, -0.868240, -1.062590, -0.574600, 0.950620, -1.275510, 1.889510, -  
0.257620, -1.017670, -1.423590,  
1.195150, 2.620400, 0.765150, 0.961900, -0.464470, 3.012470, 2.745960, -  
5.151960, -1.740690, 5.859210,  
  
0.901960, -1.685510, -6.791020, 2.203980, 5.626140, -0.384440, -1.673950,  
0.035740, 1.471390, 5.631110,  
0.421130, 0.498120, -1.716780, 3.635380, 3.566360, -0.170550, 1.744860, -  
0.654360, 1.944680, -2.919340,  
  
0.435450, -0.531890, 0.796660, -0.225860, 0.079260, -0.533410, 0.291360, -  
2.092870, 0.786100, 0.388000,  
-0.594070, 0.482830, 1.227620, -1.733030, -0.130970, -2.179940, -1.990480,  
-0.608840, -0.945180, -2.407810,  
  
11.351920, 9.836940, 0.452710, -1.368560, -1.077680, -12.011910, -8.768280,  
0.130830, 5.546830, -22.110500,  
-3.285950, 0.913410, -4.608020, -3.318360, -0.482840, -2.698960, -7.664700,  
-2.259960, -3.247980, 6.165640,  
  
1.239820, -0.004910, -0.229110, -1.199260, -0.373160, 0.404840, 0.934900,  
0.660140, 0.882870, -0.988120,  
1.272550, 1.291110, -1.026580, -3.192560, -0.530040, 1.125480, 0.550220, -  
0.605210, -1.852840, -1.936150,  
  
-2.150220, -0.901440, -1.262290, -0.574240, 0.190600, -0.876910, 0.181770,  
0.039830, -1.299620, -2.785500,  
-0.772240, 2.205490, 0.603100, -1.465930, -2.329050, -0.544750, 0.568940,  
2.268800, -1.164400, -1.476410,  
  
-2.585070, 0.478080, -3.164530, -0.497310, 1.031690, -1.191580, -3.902240,  
2.772790, 5.048490, -5.867210,  
5.380520, -0.144440, -8.988320, -4.958000, 5.619430, 9.416580, 0.734840, -  
15.394920, 1.986260, 5.200210,  
  
1.062780, -0.844970, -1.098720, -1.070040, -1.047410, 1.593420, 1.069870, -  
2.025040, 1.194330, -2.173930,  
-1.010670, -0.713490, 3.440110, 1.247560, -0.786710, -1.275810, -0.573030,  
1.814400, 2.323420, 0.873060,  
  
};
```



```

/* unit definition section (see also UnitType) */
static UnitType Units[42] =
{
    { 0.0, 0.0, 0, NULL , NULL },
    { /* unit 1 (Old: 1) */
      0.0, 0.979380, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
    { /* unit 2 (Old: 2) */
      0.0, 0.714920, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
    { /* unit 3 (Old: 3) */
      0.0, 0.193420, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
    { /* unit 4 (Old: 4) */
      0.0, -0.603470, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
    { /* unit 5 (Old: 5) */
      0.0, 0.961620, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
    { /* unit 6 (Old: 6) */
      0.0, -0.174930, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
    { /* unit 7 (Old: 7) */
      0.0, -0.971240, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
    { /* unit 8 (Old: 8) */
      0.0, 0.355700, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
    { /* unit 9 (Old: 9) */
      0.0, -0.566010, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
    { /* unit 10 (Old: 10) */
      0.0, -0.081730, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
    { /* unit 11 (Old: 11) */
      0.0, 0.976320, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
    { /* unit 12 (Old: 12) */
      0.0, 0.612350, 0,
      &Sources[0] ,
      &Weights[0] ,
    },
}

```

```
{ /* unit 13 (Old: 13) */
  0.0, -0.100210, 0,
  &Sources[0] ,
  &Weights[0] ,
},
{ /* unit 14 (Old: 14) */
  0.0, -0.490410, 0,
  &Sources[0] ,
  &Weights[0] ,
},
{ /* unit 15 (Old: 15) */
  0.0, -0.672430, 0,
  &Sources[0] ,
  &Weights[0] ,
},
{ /* unit 16 (Old: 16) */
  0.0, 0.771300, 0,
  &Sources[0] ,
  &Weights[0] ,
},
{ /* unit 17 (Old: 17) */
  0.0, 0.010670, 0,
  &Sources[0] ,
  &Weights[0] ,
},
{ /* unit 18 (Old: 18) */
  0.0, 0.085580, 0,
  &Sources[0] ,
  &Weights[0] ,
},
{ /* unit 19 (Old: 19) */
  0.0, 0.652580, 0,
  &Sources[0] ,
  &Weights[0] ,
},
{ /* unit 20 (Old: 20) */
  0.0, -0.828800, 0,
  &Sources[0] ,
  &Weights[0] ,
},
{ /* unit 21 (Old: 21) */
  0.0, 0.772550, 20,
  &Sources[0] ,
  &Weights[0] ,
},
{ /* unit 22 (Old: 22) */
  0.0, -4.022630, 20,
  &Sources[20] ,
  &Weights[20] ,
},
{ /* unit 23 (Old: 23) */
  0.0, 0.498690, 20,
  &Sources[40] ,
  &Weights[40] ,
},
{ /* unit 24 (Old: 24) */
  0.0, -1.483350, 20,
  &Sources[60] ,
  &Weights[60] ,
},
{ /* unit 25 (Old: 25) */
  0.0, 0.913140, 20,
  &Sources[80] ,
  &Weights[80] ,
```

```

    },
    { /* unit 26 (Old: 26) */
      0.0, -4.275200, 20,
      &Sources[100] ,
      &Weights[100] ,
    },
    { /* unit 27 (Old: 27) */
      0.0, 3.441210, 20,
      &Sources[120] ,
      &Weights[120] ,
    },
    { /* unit 28 (Old: 28) */
      0.0, -1.009790, 20,
      &Sources[140] ,
      &Weights[140] ,
    },
    { /* unit 29 (Old: 29) */
      0.0, -0.084640, 20,
      &Sources[160] ,
      &Weights[160] ,
    },
    { /* unit 30 (Old: 30) */
      0.0, -2.999350, 20,
      &Sources[180] ,
      &Weights[180] ,
    },
    { /* unit 31 (Old: 31) */
      0.0, -0.008810, 20,
      &Sources[200] ,
      &Weights[200] ,
    },
    { /* unit 32 (Old: 32) */
      0.0, 0.951170, 20,
      &Sources[220] ,
      &Weights[220] ,
    },
    { /* unit 33 (Old: 33) */
      0.0, -4.444250, 20,
      &Sources[240] ,
      &Weights[240] ,
    },
    { /* unit 34 (Old: 34) */
      0.0, 0.237880, 20,
      &Sources[260] ,
      &Weights[260] ,
    },
    { /* unit 35 (Old: 35) */
      0.0, -1.782990, 20,
      &Sources[280] ,
      &Weights[280] ,
    },
    { /* unit 36 (Old: 36) */
      0.0, 1.067750, 20,
      &Sources[300] ,
      &Weights[300] ,
    },
    { /* unit 37 (Old: 37) */
      0.0, 1.076860, 20,
      &Sources[320] ,
      &Weights[320] ,
    },
    { /* unit 38 (Old: 38) */
      0.0, -3.405790, 20,
      &Sources[340] ,

```



```

        &Weights[340] ,
    },
    { /* unit 39 (Old: 39) */
        0.0, -3.614690, 20,
        &Sources[360] ,
        &Weights[360] ,
    },
    { /* unit 40 (Old: 40) */
        0.0, 8.839670, 20,
        &Sources[380] ,
        &Weights[380] ,
    },
    { /* unit 41 (Old: 41) */
        0.0, 0.665960, 20,
        &Sources[400] ,
        &Weights[400] ,
    }
};

```

```
int SNNS20d(float *in, float *out, int init)
```

```

{
    int member, source;
    float sum;
    enum{OK, Error, Not_Valid};
    pUnit unit;

    /* layer definition section (names & member units) */

    static pUnit Input[20] = {Units + 1, Units + 2, Units + 3, Units + 4,
        Units + 5, Units + 6, Units + 7, Units + 8, Units + 9, Units + 10, Units +
        11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16, Units + 17,
        Units + 18, Units + 19, Units + 20}; /* members */

    static pUnit Hidden1[20] = {Units + 21, Units + 22, Units + 23, Units +
        24, Units + 25, Units + 26, Units + 27, Units + 28, Units + 29, Units + 30,
        Units + 31, Units + 32, Units + 33, Units + 34, Units + 35, Units + 36,
        Units + 37, Units + 38, Units + 39, Units + 40}; /* members */

    static pUnit Output1[1] = {Units + 41}; /* members */

    static int Output[1] = {41};

    for(member = 0; member < 20; member++) {
        Input[member]->act = in[member];
    }

    for (member = 0; member < 20; member++) {
        unit = Hidden1[member];
        sum = 0.0;
        for (source = 0; source < unit->NoOfSources; source++) {
            sum += unit->sources[source]->act
                * unit->weights[source];
        }
        unit->act = Act_Logistic(sum, unit->Bias);
    };

    for (member = 0; member < 1; member++) {
        unit = Output1[member];
        sum = 0.0;
        for (source = 0; source < unit->NoOfSources; source++) {

```

```

        sum += unit->sources[source]->act
              * unit->weights[source];
    }
    unit->act = Act_TanH_Xdiv2(sum, unit->Bias);
};

for(member = 0; member < 1; member++) {
    out[member] = Units[Output[member]].act;
}

return(OK);
}

```

B.4 Gyro characterisation: Gyro2.PAS (Pascal)

```

program gyro2;
uses
    DOS, GRAPH, CRT;
const
    PERIODE = 1;
    ADC      = $360;           {Addresses of ADDAS card}
    ADChi    = $362;
    DAC0     = $364;
    DAC1     = $366;
    {filename = 'd:\teststep.dat';}
    {writefile = 'd:\ADOUT.dat';}

type
    lintfiletype = file of longint;
    wordfiletype = file of word;

var
    VLAG          : boolean;
    ADDATA        : array[0..3] of integer;
    VEKTOR        : pointer;
    segarray, offarray, segarrayw, offarrayw : array [0..7] of word;
    nn, mm, test  : word;
    precessfile   : lintfiletype;
    lambdafile    : wordfiletype;
    p1, p2, p3, p4, p5, p6, p7, p8, w1, w2, w3, w4, w5, w6, w7, w8 : ^byte;
    lambdaname, precessname : string;
    precess, intcounter : longint;
    precess1, argument, zmin1, zmin2, swingrate : real;

procedure AtoD;
begin
    inline(
        $BA/$60/$03/ {mov dx, ADC}
        $EC/          {in al, dx} {Dummy Read}
        $EE/          {out dx, al} {Start conversion}
        $B9/$00/$01/  {mov cx, 100H}
        $BA/$62/$03/  {mov dx, ADCHI}
        {@1:}
        $EE/          {out dx, al} {Clock interrupt line in
latch}
        $EC/          {in al, dx} {Poll ADC interrupt}
        $24/$80/      {and al, 80H}
        $E0/$FA/      {loopnz @1}
        $BF/ADDATA/    {mov di, offset ADDATA}
        $B9/$04/$00/   {mov cx, 2}
        {@2:}
        $BA/$60/$03/  {mov dx, ADC}
    )
end

```

```

$EC/          {in al,dx}          {Read low byte}
$88/$C3/      {mov bl,al}
$BA/$62/$03/  {mov dx,ADCHi}
$EC/          {in al,dx}          {Read high byte}
$24/$0F/      {and al,0FH}
$88/$C7/      {mov bh,al}
$24/$08/      {and al,08H}
$74/$04/      {jz @3}
$81/$EB/$00/$10/ {sub bx,1000H}
              { @3: }
$89/$1D/      {mov [di],bx}       {Store channel A/D data}
$47/          {inc di}
$47/          {inc di}
$E2/$E4);    {loop @2}
end; {AtoD}

```

```

procedure DtoA(K,SS : longint);    {DA PROCEDURE}
var
  S : integer;
begin
  S:=integer(SS);
  if S > 2047 then S := 2047;
  if S < -2048 then S := -2048;
  S := (S + 2048) shl 4;
  if K = 0 then begin
    port[DAC0+1] := hi(S);
    port[DAC0] := lo(S);
  end
  else begin
    port[DAC1+1] := hi(S);
    port[DAC1] := lo(S);
  end;
end; {DtoA}

```

```

procedure KLOK; assembler;
asm
  push  ax
  push  ds
  mov   ax, seg @DATA
  mov   ds, ax
  mov   VLAG, 1
  mov   al, 20H
  out   20H, al
  pop   ds
  pop   ax
  iret
end;

```

```

procedure INIT;
var
  II : byte;
begin
  getintvec($08,VEKTOR);
  port[$43]:=$36;
  port[$40]:=lo(238);    {50ms=59659, 20ms=23864}
  port[$40]:=hi(238);
  setintvec($08,@KLOK);
end;

```

```

procedure STOP;
begin
  port[$43]:=$36;
  port[$40]:=0;

```



```

    port[$40]:=0;
    setintvec($08,VEKTOR);
    DtoA(0,0);
end;

{HOOFFPROGRAM}
begin
    write(' Precess Swing Rate (Hz): ');
    readln(swingrate);

    write(' Precess Data Filename: ');
    readln(precessname);

    write(' Lambda Coil Data Filename: ');
    readln(lambdaname);
    {delay(1000);}

    writeln('>> Allocating DA Memory <<');
    getmem(p1,41500);
    getmem(p2,41500);
    getmem(p3,41500);
    getmem(p4,41500);
    getmem(p5,41500);
    getmem(p6,41500);
    getmem(p7,41500);
    getmem(p8,41500);

    segarray[0]:=seg(p1^);
    segarray[1]:=seg(p2^);
    segarray[2]:=seg(p3^);
    segarray[3]:=seg(p4^);
    segarray[4]:=seg(p5^);
    segarray[5]:=seg(p6^);
    segarray[6]:=seg(p7^);
    segarray[7]:=seg(p8^);

    offarray[0]:=ofs(p1^);
    offarray[1]:=ofs(p2^);
    offarray[2]:=ofs(p3^);
    offarray[3]:=ofs(p4^);
    offarray[4]:=ofs(p5^);
    offarray[5]:=ofs(p6^);
    offarray[6]:=ofs(p7^);
    offarray[7]:=ofs(p8^);

    writeln('>> Allocating AD Memory <<');
    getmem(w1,20750);
    getmem(w2,20750);
    getmem(w3,20750);
    getmem(w4,20750);
    getmem(w5,20750);
    getmem(w6,20750);
    getmem(w7,20750);
    getmem(w8,20750);

    segarrayw[0]:=seg(w1^);
    segarrayw[1]:=seg(w2^);
    segarrayw[2]:=seg(w3^);
    segarrayw[3]:=seg(w4^);
    segarrayw[4]:=seg(w5^);
    segarrayw[5]:=seg(w6^);
    segarrayw[6]:=seg(w7^);
    segarrayw[7]:=seg(w8^);

```

```

offarrayw[0]:=ofs(w1^);
offarrayw[1]:=ofs(w2^);
offarrayw[2]:=ofs(w3^);
offarrayw[3]:=ofs(w4^);
offarrayw[4]:=ofs(w5^);
offarrayw[5]:=ofs(w6^);
offarrayw[6]:=ofs(w7^);
offarrayw[7]:=ofs(w8^);

writeln('>> Reading DA File <<');
assign(lambdofile,lambdaname);
rewrite(lambdofile);
assign(precessfile,precessname);
rewrite(precessfile);
(*
for nn:=0 to 7 do
begin
write('          Segment: ');
write(nn+1);
write(' ');
writeln(' ');
for mm:=0 to 10374 do
begin
read(datafile,meml[segarray[nn]:(offarray[nn]+4*mm)]);
end;
end;*)

writeln('>> Init IRQ 8 <<');
INIT;
writeln('>> Start Conversion <<');

intcounter:=0;
precess:=0;
zmin1:=0.0;
zmin2:=0.0;
for mm:=0 to 7 do
begin
for nn:=0 to 10374 do
begin
repeat until VLAG;           {Wait for interrupt}
VLAG:=false;
intcounter:=intcounter+1;
AtoD;
DtoA(0,precess);
{AtoD;}
{precess:=ADDA[0];}
precess1:=zmin2;
zmin2:=zmin1;
{argument:=(intcounter/5000);}
zmin1:=-
0.5*zmin2+1.3696*zmin1+(0.1*ADDA[0]*sin(swingrate*6.283185307*intcounter/
5000));      {filter and AM generation}
precess:=trunc(precess1);
{DtoA(0,precess);}
(* DtoA(0,meml[segarray[mm]:(offarray[mm]+4*nn)]);      {-2048 to +2047 =
-10 to +10 Volt}*)
memw[segarrayw[mm]:(offarrayw[mm]+2*nn)]:=word(ADDA[1]+32768);
meml[segarray[mm]:(offarray[mm]+4*nn)]:=precess;
end;
end;
(*writeln('>> DA Conversion Completed <<');
writeln('>> Freeing DA Memory <<');*)
STOP;

```

```
writeln('>> Saving AD Data <<');
for nn:=0 to 7 do
begin
  write('      Save Segment: ');
  write(nn+1);
  write(' ');
  writeln(' ');
  for mm:=0 to 10374 do
  begin
    write(lambdafile,memw[segarrayw[nn]:(offarrayw[nn]+2*mm)]);
    write(processingfile,meml[segarray[nn]:(offarray[nn]+4*mm)]);
  end;
end;
writeln('>> Freeing AD Memory <<');
freemem(w1,20750);
freemem(w2,20750);
freemem(w3,20750);
freemem(w4,20750);
freemem(w5,20750);
freemem(w6,20750);
freemem(w7,20750);
freemem(w8,20750);

freemem(p1,41500);
freemem(p2,41500);
freemem(p3,41500);
freemem(p4,41500);
freemem(p5,41500);
freemem(p6,41500);
freemem(p7,41500);
freemem(p8,41500);

close(processingfile);
close(lambdafile);

writeln('>> All Stop Halt etc.... <<');

end. {gyro2}
```


APPENDIX C: A/D AND D/A CONVERTER MODELLING

C.1 A/D converter response measurement

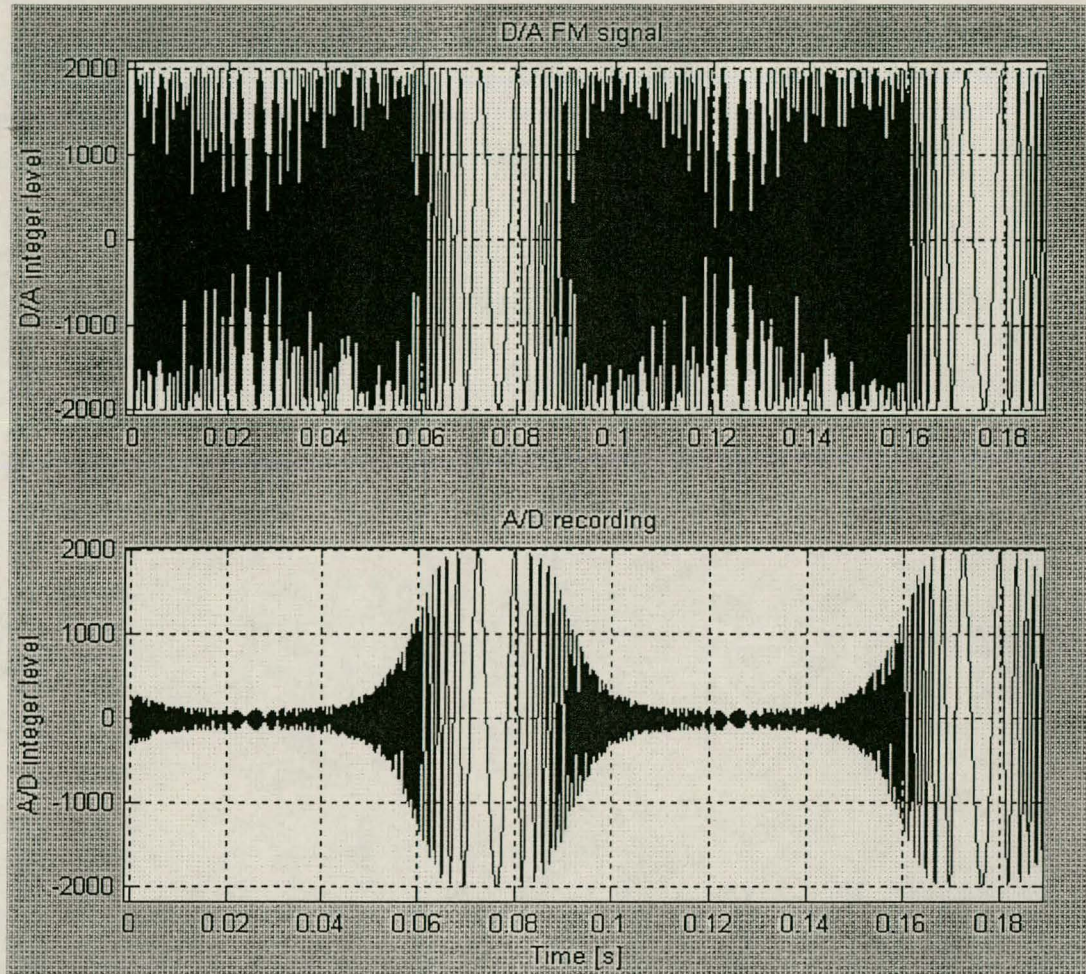


Figure 46 A/D converter response measurement

Making use of Matlab, a wide band FM signal is created. The signal contains almost an equal frequency spread between 50Hz and 5kHz, and is illustrated in the upper half of Figure 46. This frequency band was chosen because it is known that the anti-aliasing filter (AAF) of the A/D converter is a LPF with a corner frequency in the vicinity of 1000Hz. The wide band FM signal is written to file. A software program much like Gyro2.PAS in Appendix B, is used to write the signal from the file, out through the D/A converter port of the ADAS-card. The D/A port is modelled by a ZOH circuit. An A/D-port is connected directly to the D/A output, recording the D/A signal. The A/D recording passes through the anti-aliasing filter, which is the only relevant distortion influencing the recording. Additional effects such as discrete signal quantisation and A/D buffer slew rate ability do influence the signal, but to a lesser degree. The

amplitude of the D/A signal is approximately 20Vpp, decreasing the relative quantisation error to a minimum, while increasing the risk of exceeding the slew rate constraints. However, the frequency content of the signal is low enough to ensure remaining within the slew rate limitation envelope.

C.2 A/D converter modelling

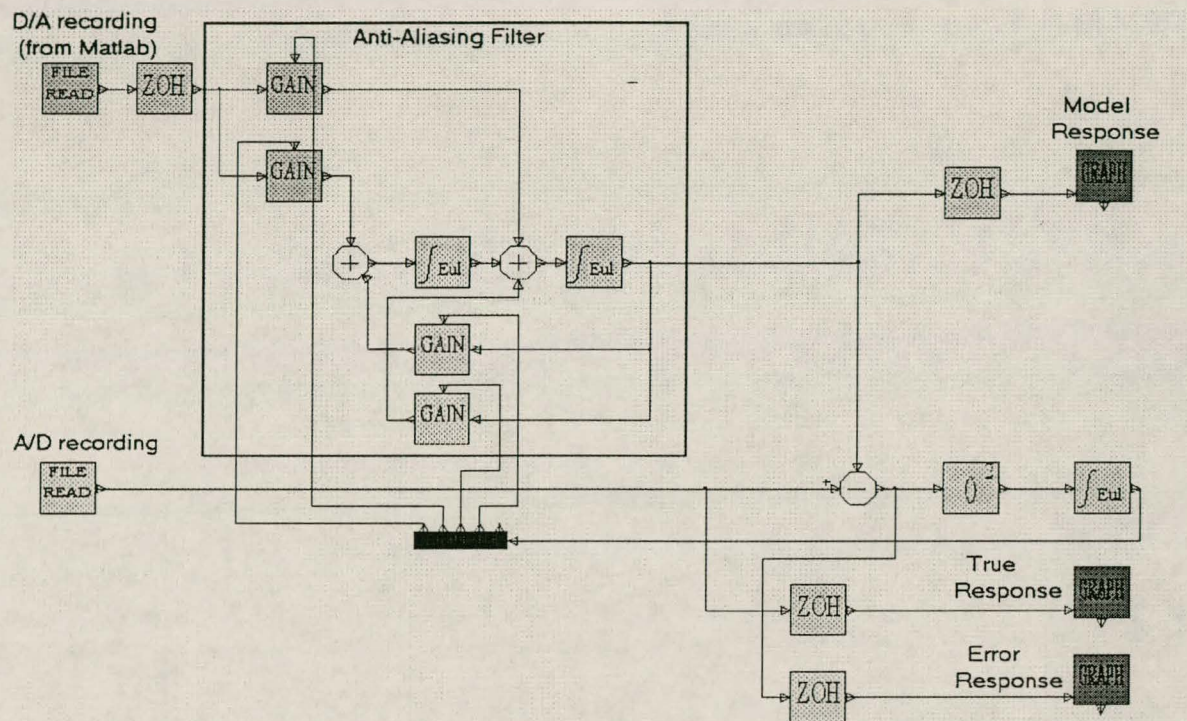


Figure 47 A/D converter model optimisation

Figure 47 illustrates the A/D converter model, consisting solely of the AAF and its surrounding optimisation routine. The AAF is a second order LPF section, resulting in a second order section being built into the model. This optimisation routine optimises the model to a mean A/D integer level error of -4.8 . Relative to the 4000 levels of discretisation used when representing the peak-to-peak A/D signal, this mean error becomes insubstantial. When the model response is plotted together with the original FM D/A signal, almost no discernible difference exists. The AAF transfer function optimises to:

$$G(s) = \frac{-455.934s + 50.863 \times 10^6}{s^2 + 10.851 \times 10^3 + 50.632 \times 10^6} \quad \text{Equation 6}$$

This transfer function results in a non-minimum phase LPF with -3dB corner frequency at 1048Hz.